CONTROL DATA® 6000 SERIES COMPUTER SYSTEMS

CHIPPEWA OPERATING SYSTEM DOCUMENTATION

Volume III   Preliminary Edition

CONTROL DATA CORPORATION

Development Division - Applications

DSD - THE SYSTEM DISPLAY

Chippewa Operating System

11/25/65

DSD: The System Display

Introduction

The display console is controlled by a display program, DSD, which permanently resides in peripheral processor 9. DSD displays a variety of information concerning the status of the system, including a display of the dayfile, a display of the jobs waiting to be executed and waiting to be printed, and a display showing the status of each control point. DSD permits selected portions of central memory to be displayed, and also provides for the modification of central memory locations. In addition to its display function, DSD processes keyboard messages from the operator. Operator functions include bringing a job to or dropping a job from a control point, assigning equipment, and selection of various types of displays. The Chippewa Operating System also contains a job display package, DIS. DIS, when called, is assigned to a control point, and permits the modification of job parameters, memory locations, and control statement sequences for this job assigned to the control point. For the most part, DSD and DIS displays are identical.

The main components of the display console are the two cathode ray tubes and the keyboard. By issuing the appropriate function codes to the display console controller, displays of 16, 32, or 64 characters per line may be selected on either the right or left screens. A dot mode display is also available, although only the character mode display is used by the operating system. The display area can be considered to be composed of a grid of points, 512 by 512 points in size. A display can be initiated at any point in the display area by issuing the coordinates of that point. A vertical, or Y, coordinate is sent to the controller in the low-order nine bits of a byte in which the high-order octal digit is a 7. Similarly, a horizontal, or X, coordinate is sent to the controller in the low-order nine bits of a byte in which the high-order octal digit is a 6. If the display console controller receives a byte in which the high-order octal digit is neither a 6 or a 7, it is assumed that this byte contains two display code characters.

To display a line of information on the screen, an X and a Y coordinate are sent to the controller via the appropriate output instructions

(OAN or OAM). These coordinates define the location of the lower left corner of the first character to be displayed. The information to be displayed is then sent to the controller via an OAM instruction. As each character is displayed the X coordinate is automatically incremented. To display another line, the X and Y coordinates should be reinitialized. A coordinate of X=000 defines the left-most boundary of the display area; a coordinate of $X=777_8$ defines the right-most boundary of the display area; a coordinate of $Y=777_8$ defines the upper boundary of the display area, and a coordinate of Y=000 defines the lower boundary of the display area.

The Chippewa Operating System uses a display of 64 characters per line in both DSD and DIS. Generally, the Y (vertical) coordinate spacing between successive lines is 12_8. The display must be regenerated at least 25 times per second in order to avoid flicker. The DSD display is designed to maintain an average rate of 40 displays per second.

DSD Master Loop

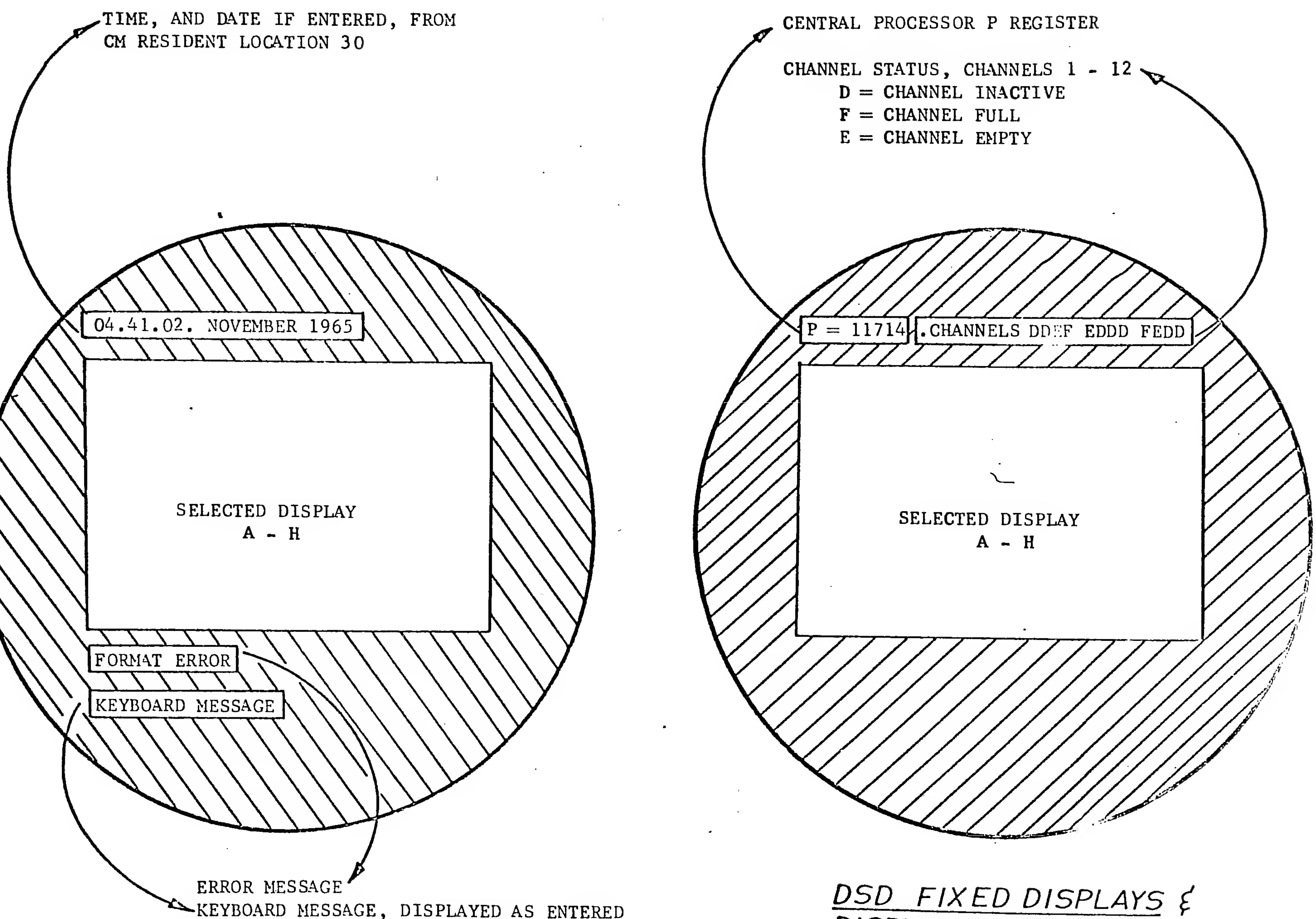
The DSD master loop is shown on page A-1 of the attached flow charts. On the initial entry to this routine (i.e., at dead start time), a subroutine is called to perform housekeeping. This subroutine clears the temporary storage areas used by DSD, selects the "A" display (dayfile) on the left screen and the "B" display (control points) on the right screen, and requests reservation of channel 10 from MTR. Display selection in DSD is performed by setting the address of the desired display subroutine in location 70 for the left screen and in location 71 for the right screen.

On each pass through its master loop, DSD selects the display console keyboard and issues an input instruction to read the keyboard. If a zero byte is returned, then no key has been depressed since the last pass through the master loop. If a non-zero byte is returned, the keyboard character in the low-order bits of the byte is processed. If the character is a carriage return, the Message Ready flag is set and the keyboard message is processed.

After keyboard processing is completed, DSD issues a function code to select the left screen, and displays the time and date from central memory resident beginning at location 30. The contents of this area are read and displayed, word by word, until a zero byte is encountered. Regardless of the display selected for the left screen, the time-date line is always displayed. DSD then jumps to the subroutine whose address is contained in location 70 to process the selected left screen display. At the bottom of the left screen, the keyboard message currently being entered is displayed. If an error is encountered in processing this message, the error message "FORMAT ERROR" will be displayed immediately above the keyboard message. Once processing of a valid message is complete, the message will be no longer displayed.

DSD then issues a function code to display the right screen. At the top of the right screen, the contents of the central processor P register and the status of the 12 data channels are always displayed. DSD reads the central processor P register, converts the contents of the P register to display code, and displays these characters. On the same line, three groups of four characters, one character for each of the 12 data channels, are displayed. Each channel is first tested to determine if it is active or inactive. If the channel is inactive, the displayed character corresponding to that channel is a "D" (disconnected). If the channel is active and empty, an "E" is displayed, while if the channel is active and full, an "F" is displayed. (See figure 1) DSD then jumps to the subroutine whose address is contained in location 71 to process the selected right screen display.

After both screens have been displayed, DSD calls the Adjust Display Period subroutine (shown on page A-1 of the attached flow charts). The purpose of this subroutine is to control the number of passes made through DSD's master loop in a fixed time interval in order to avoid flicker. On each entry to the Adjust Display Period subroutine (i.e., on each pass through DSD's master loop), a display cycle counter is advanced. At the end of each second, the display cycle count is examined to determine if the display was repeated more than 50_8 times in the past second. If it was, a delay count (D) equal to the display cycle count - 50_8 is set. If the display was repeated less than 50 times in the past second, this delay count is set to zero. The delay count is used to establish



DSD FIXED DISPLAYS &
DISPLAY PLACEMENT

Figure 1

a delay between successive passes through DSD's master loop: the larger the delay, the greater the time between successive passes through the master loop. Also, in the next second, D will be set to zero if the display is repeated 50-D times or less. If in any second, then, the display is repeated more than 50_8 times, a delay so that the display will be repeated less than 50_8 times in the subsequent second: over a period of several seconds, the display rate should average out to 50_8 times per second. At the end of each second, the display cycle counter is reset to zero.

In a 6000 system with a single display console, DSD must relinquish control to DIS when the latter is called to a control point. When DIS begins execution, it requests MTR to assign an equipment of type DS. MTR searches the EST for an entry of this type, and, when found, enters the requestor's control point address in byte one of the EST entry. On each pass through its master loop, DSD reads the EST entry for equipment number 10. If byte one is non-zero, then this equipment has been assigned to another user - DIS. DSD then releases the channel reservation for the channel to which the display controller is connected, and loops on a test of byte one of the EST entry. When this byte becomes zero once again, DIS has released control of the console, and so DSD requests the channel once again and returns to its master loop.

One of the keyboard entries processed by DSD is the "STEP." message, which causes MTR to enter a step mode of operation. In step mode, MTR pauses for operation intervention before processing each request from a peripheral processor. To process the STEP message, DSD sends function request 5 to MTR. MTR then sets a switch in the subroutine which processes requests from peripheral processors. When a request is next received from a peripheral processor, MTR will set a Wait flag in byte 5 of central memory location 14, and will then loop until this flag has been modified. Entering a space on the keyboard will result in the clearing of this flag by DSD: MTR will then process the request just received, but will pause again before processing subsequent requests. Entering a period on the keyboard will result in this flag's being set to 7777 by DSD: MTR will then reset the switch in the subroutine which processes peripheral processor requests, and will process subsequent requests in the normal manner. Since a space or a period is not, in

itself, a conventional DSD message, DSD checks for the entry of these characters on each pass through its master loop if MTR is in step mode.

DSD Keyboard Message Processing

The processing of characters received from the keyboard is shown in the flow chart on page A-1. If the character received is a carriage return, then a complete message has been entered and so the Message Ready flag is set. If the character received is a backspace, DSD clears the last character entered in the buffer, resets the buffer address accordingly, and clears the error flag which may have been set if an attempt was made to process the message. If the drop key was depressed, then the entire message is deleted: the buffer address is reset to the starting address, and the error flag cleared. Should the character be a valid keyboard character, it is entered in the message buffer and the buffer address advanced. Note that the space character from the keyboard (62_8) is not a display code character, and so a blank (55_8) is substituted for it.

When DSD detects that a carriage return has been entered on the keyboard, the Message Ready flag is set to indicate that a message is ready for processing. DSD then proceeds to interpret the message. Message processing is illustrated in the flow chart on page A-2. The second character is examined to determine if it is a period: if it is, then the message is a control point message, and so the first character is examined to determine if it is a valid control point number (1-7). If the first character is not a numeric in the range 1-7, the Message Error flag is set and control returned to DSD's master loop, where the message "FORMAT ERROR" will be displayed. If the first character is a valid control point number, then a table search is made for the address of the appropriate subroutine. If the message is not found in the table, the Message Error flag is set and control is returned to DSD's master loop. Processing of the valid control point messages is described below.

ONSW: If the message is of the form n.ONSWx., DSD sets the bit corresponding to X+5 in byte 5 of location RA for control point n, and in word 26 of control point area n. Control is then returned to DSD's master loop.

OFFSW: If the message is of the form n.OFFSWx., DSD clears the bit corresponding to X+5 in byte 5 of location RA for control point n, and in word 26 of control point area n. Control is then returned to DSD's master loop.

LOAD: If the message is of the form n.LOAD., DSD writes the package name (1LT) and the control point number, n, in its Message Buffer. Word 21 of control point area n is then examined to determine if the control point area contains a job name. If it does not, DSD requests MTR to assign a pool processor to control point n. MTR will copy the contents of DSD's Message Buffer into the Input Register of a free pool processor, and assign the processor to control point n. Control is then returned to DSD's master loop. If the control point area contains a job name (word 21 non-zero), control is returned to the DSD master loop. If the control point area contains a job name (word 21 non-zero), control is returned to the DSD master loop without requesting the assignment of a processor.

NEXT: Processing of the message n.NEXT. is identical to the processing of the LOAD message with the exception that the package name 1BJ is written in DSD's Message Buffer.

READ: Processing of the message n.READ. is identical to the processing of the LOAD message with the exception that the package name 1LJ is written in DSD's Message Buffer.

PRINT: Processing of the message n.PRINT. is identical to the processing of the LOAD message with the exception that the package name 1DJ is written in DSD's Message Buffer.

DIS: If the message is of the form n.DIS., DSD writes the package name DIS in its Message Buffer, and requests MTR to assign a pool processor to control point n. Control is then returned to DSD's master loop.

ASSIGN: The ASSIGN message is generally entered in response to a REQUEST statement display or the message WAITING FOR XX. If the message is of the form n.ASSIGNXX., DSD requests MTR to assign the

specified equipment to the control point. MTR looks up the corresponding entry in the EST: if equipment XX is not already assigned, then MTR assigns the equipment to control point n and writes the equipment number in word 22 in the control point area. After initiating the MTR request, control is returned to DSD's master loop.

GO: The 2^0 bit in byte 4 of location RA is a pause bit. This bit is set by a FORTRAN PAUSE statement, and is also set by certain peripheral packages when an error is detected. For example, 2RT sets this bit when a parity error has occurred after reading a record three times. When the n.GO. statement is processed, this bit is cleared. Also, the most recent message in the control point area (presumably the PAUSE statement) is cleared.

END: If the message is of the form n.END1., n.END2., n.END3., or n.END4., DSD sets a printer stop code in byte 2 of word 20 in the control point area. This stop code is equivalent to setting the low-order second octal digit of this byte to the digit following the word END in the message. The printer stop code is sensed by the four-printer print programs.

DROP: If the message is of the form n.DROP., DSD writes the control point number n in its Output Register, and requests MTR to drop the job at control point n. MTR sets error flag six (Operator Drop) to initiate error processing.

If the second character in the message was a period, and the message was not found to be one of those described above, the Message Error flag is set and control returned to DSD's master loop. If the second character in the message was not a period, the first character is examined to determine if it is an octal digit. Should the first character be an octal digit, it is assumed that the message is a storage entry message of the form a,d., where a represents a central memory address and d represents the data to be entered in memory at that address. The characters in the message are assembled and converted to octal until a separator is found: if the separator is not a comma, the Message Error flag is set. Once the address has been assembled, the characters following the comma

are assembled and converted to octal until another separator is found. If this separator is not a period, the Message Error flag is set. The assembled data is stored, right-justified, in a 5-byte area, and the contents of this area are then written in central memory at the specified address.

If the first character is not an octal digit, the third character is examined to determine if it is a period. If the third character is a period, the message is assumed to be a display mode message of the form AB., where A and B represent characters specifying the desired display on the left and right screens respectively. The subroutine address corresponding to the specified display (A-H) is located in a table and stored in location 70, in the case of the left screen display, or location 71, in the case of the right screen display.

If the third character was not a period but was a comma, it is assumed that the message is a display field change message of the form mf,a., where m is the display mode (C-G), A is the field whose starting address is to be changed (0-3 for fields 0-3, or 4 for all four fields), and a is the new starting address. Each of the storage display subroutines for storage displays C, D, E, F, and G maintains a list of four addresses, one for each of the four fields displayed. When this message is detected, DSD modifies the appropriate address in the list of field addresses for the specified display if the second character is 0-3. If the second character of the message is 4, the first address in the list is set to the address contained in the message, the second address in the list is set to the address contained in the message plus 10_8, and so forth.

If the third character in the message is not a comma, it is assumed to be a non-control point message of the form described below. DSD searches a table for the address of the appropriate subroutine: if the message is not found in the table, the Message Error flag is set and control returned to DSD's master loop. Processing of these messages is described below.

<u>DCN</u>: If the message is of the form DCNXX., where XX is an octal channel number, DSD assembles the channel number and tests the channel to determine if it is inactive. If the channel is active,

a channel disconnect is issued.

FCN: If the message is of the form FCNXX., where X is an octal channel number, the channel number is assembled and a test made to determine if the channel is inactive. If the channel is inactive, a zero function is sent to the channel.

AUTO: If the message is AUTO., DSD assigns READ (1LJ) to control point 1, PRINT (1DJ) to control point 2, and NEXT (1BJ) to control points 3, 4, 5, and 6. To assign a package to a control point, DSD writes the package name and control point number in bytes one and two of word one of its Message Buffer, and sends function request 20, Assign PPU, to MTR. MTR locates a free pool processor, assigns it to the control point, and copies word one of DSD's Message Buffer into the pool processor's Input Register.

STEP: If the message is STEP., DSD requests MTR to enter step mode by sending function request 5 to MTR. (See discussion on page 5.)

ON or OFF: The messages ONXX. and OFFXX. permit the operator to clear and set, respectively, the interlock bit in the EST table entries. In these messages, XX is an octal equipment number which defines a location in EST. The interlock bit is generally used only with magnetic tape units. When this bit is set, the corresponding equipment will not be automatically allocated in response to an ASSIGN request: if this bit is cleared, and a request such as ASSIGN MT is processed, the equipment will be automatically assigned by MTR. For equipment types MT and WT, this bit is set at load time. (See page 9 of CENTRAL MEMORY RESIDENT.)

TIME: If the message begins with the characters TIME., and contents of the keyboard message buffer following these characters are copied into central memory resident beginning at location 30. This information may comprise up to six central memory words. It is assumed that the first portion of this message has the form _HR.MN. SC., where HR represents hours, MN represents minutes, and SC represents seconds. This time will be advanced by MTR and will

appear in all dayfile messages and at the top of the left screen display. The information following the time may be the date and/or any other desired information. The date portion will also be displayed at the top of the left screen, and will be printed at the end of a job's dayfile listing.

DSD Displays

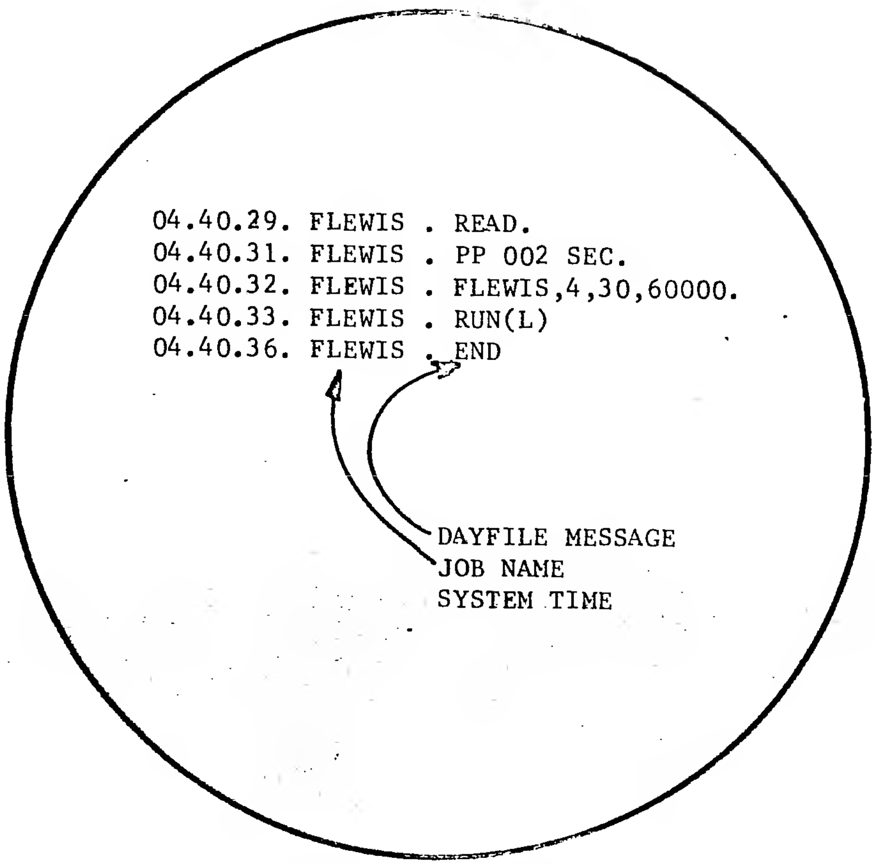
The various display modes which may be selected in DSD are as follows:

A.........Dayfile Display
B.........Control Point Display
C.........Storage Display (5 groups of 4 digits)
D.........Storage Display (5 groups of 4 digits)
E.........Storage Display (5 groups of 4 digits)
F.........Storage Display (4 groups of 5 digits)
G.........Storage Display (4 groups of 5 digits)
H.........Job Backlog Display

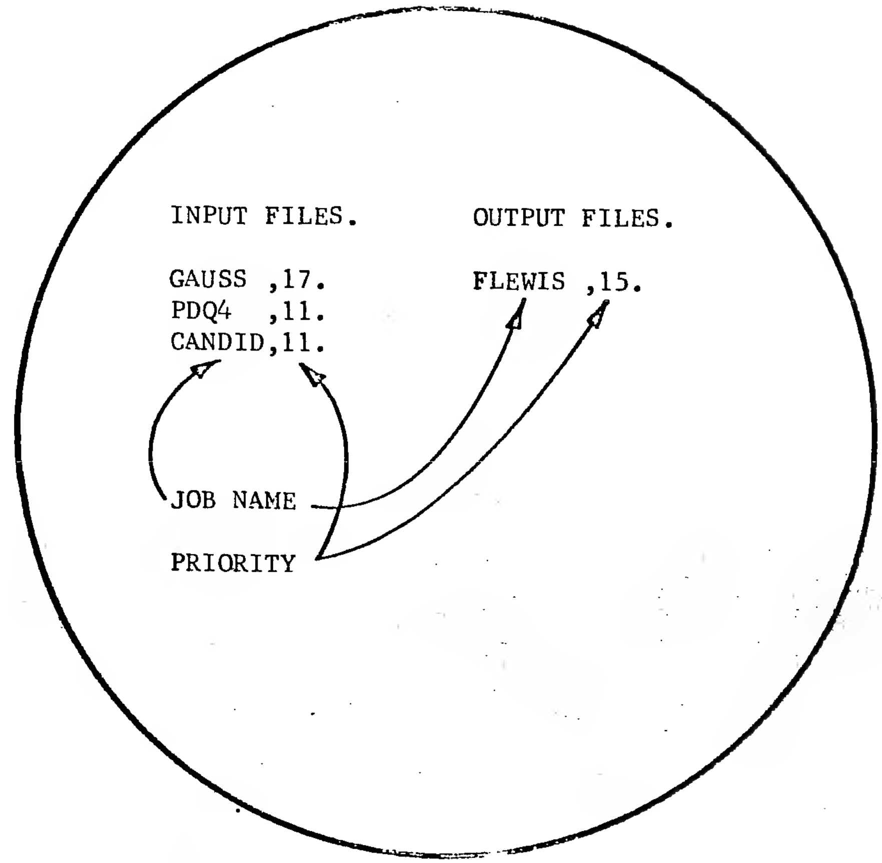
The format of these displays is illustrated in figures 2, 3, and 4. Each display is processed by a separate DSD subroutine: these subroutines appear on pages A-3 and A-4 of the attached flow charts. For the most part, display processing is quite straightforward, and discussions of these displays will be limited to points of interest.

"A" Display: The "A" display is a display of the dayfile buffer (DFB) contents from FIRST to IN. It is possible that the bottom of the display area may be reached before all the information in the dayfile has been displayed. If so, the subroutine parameters are modified so that on the next entry to the subroutine, the message which previously appeared at the top of the display will not be processed, thus permitting a new message to be displayed. Also, the point at which the display begins is moved down by the width of a line, and gradually moved back up during the next 10 displays. As a result, a revolving or rolling effect is obtained.

"B" Display: The B display shows information concerning each of the seven control points, as shown in figure 3. On entry to this subroutine, the copy of the control stack which MTR maintains in locations 56-57 of central memory resident is read. For each control



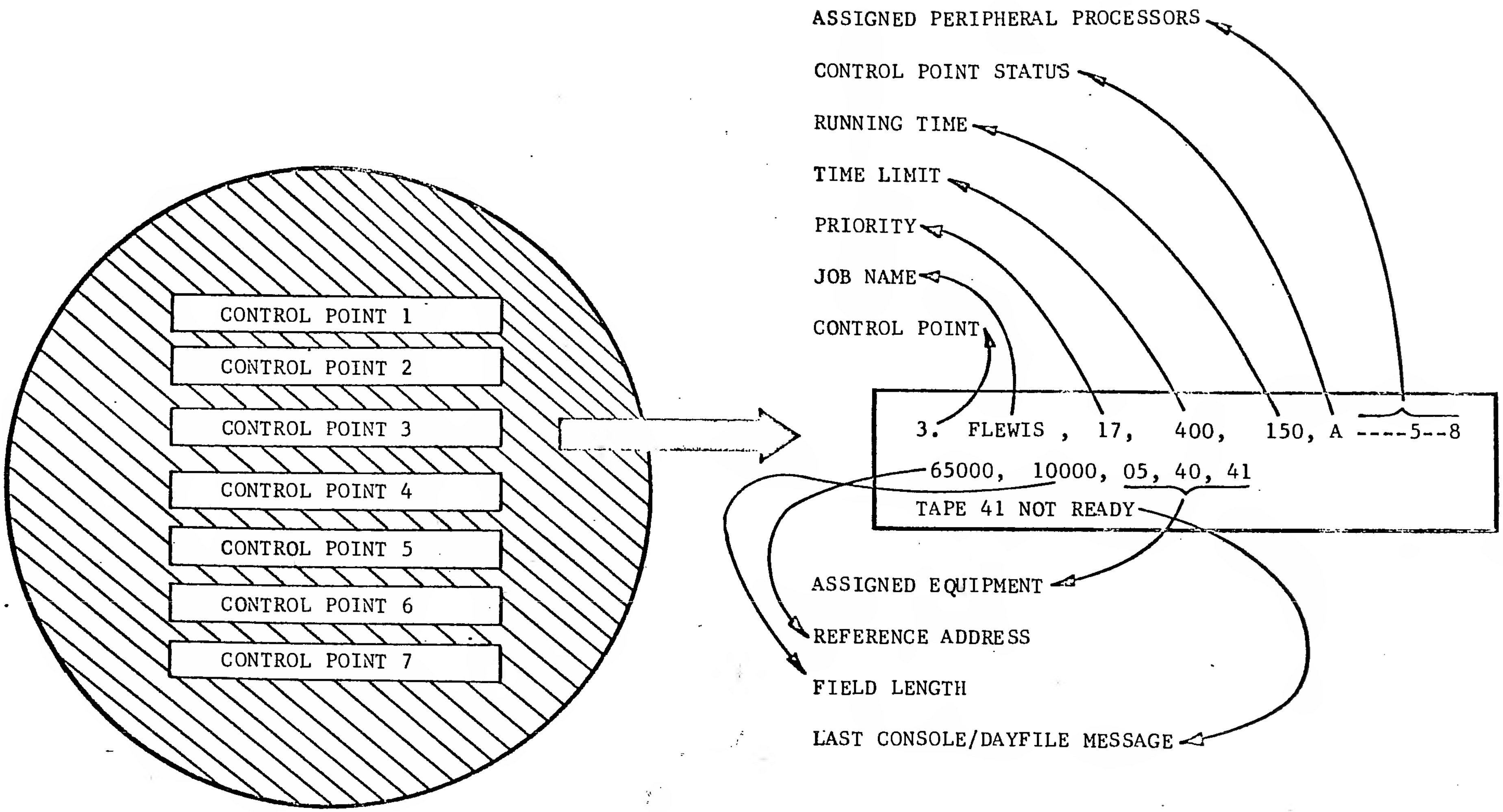
DISPLAY "A" - DAYFILE



DISPLAY "H" - JOB BACKLOG

DSD A & H DISPLAYS

Figure 2



DSD "B" DISPLAY: CONTROL POINTS

Figure 3

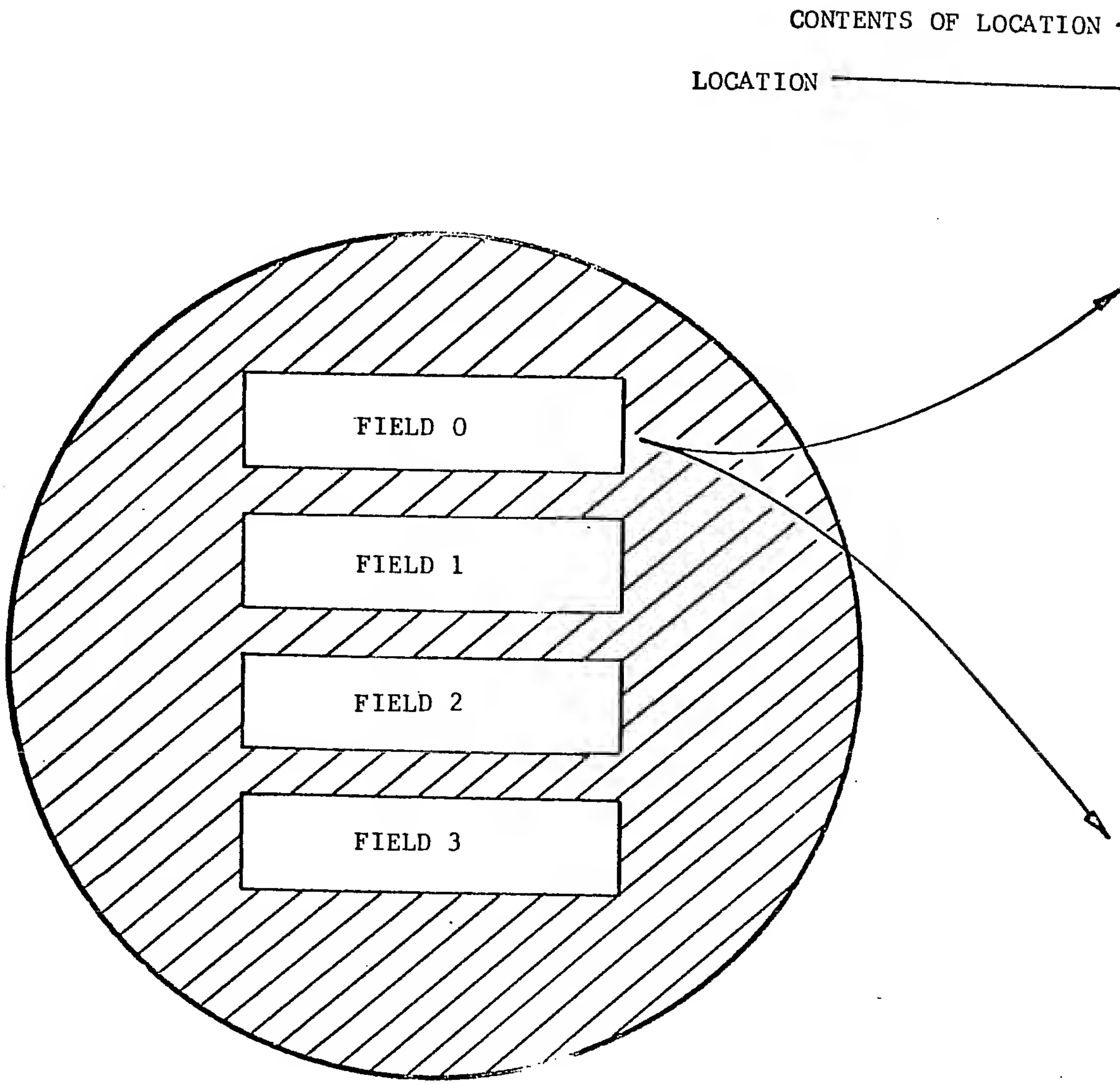
point in the stack, a status indicator, A-G, is set to represent the position of that control point in the stack (i.e., A represents the top of the stack, B the second entry in the stack, and so forth). The status byte in each control point is also read, and the status indicator set to W or X depending on the setting of these flags. The remaining processing performed by the subroutine consists of reading information from the control point area and displaying this information.

<u>Displays C-G</u>: The storage displays, C through G, each display 4 fields of 8 central memory words (see figure 4). Displays C, D, E are identical in format and display each central memory word as 5 groups of 4 octal digits. Displays F and G are identical in format and display each central memory word as 4 groups of 5 octal digits. There is a separate subroutine for each of these five displays: each of these subroutines maintains a list of four field addresses which specify the starting point for each of the four 8-word field displays. These address lists are set via keyboard messages (see page 9). The address lists are initialized at load time as follows:

C Display .....Words 20-27 of control point areas 1, 2, 3, and 4
D Display .....CM resident locations 0-37
E Display .....CM resident locations 60-117
F Display .....Central Memory locations 10000-10037
G Display .....Central Memory locations 10040-10077

The reason for identical displays (C, D, E, and F, G) is to permit the operation to switch between scans of selected memory areas without the necessity of entering starting addresses each time he switches from one area to another.

<u>"H" Display</u>: The H display lists the input and output files in the FNT. Upon entry to the H display subroutine, the FNT is searched and two lists prepared: one, a list of FNT addresses for entries of file type INPUT, and the other a list of FNT addresses for entries of file type OUTPUT. The entries represented in these lists



04600	00000	00000	00000	00000
04601	00000	00000	00000	01000
04602	51202	60002	63220	43052
04603	67312	51300	00002	11730
04604	56730	51402	60003	10740
04605	51700	00002	71730	00003
04606	76710	51730	00004	67112
04607	61102	60005	01004	00200

4 GROUPS OF 5 OCTAL DIGITS
(DISPLAYS F AND G)

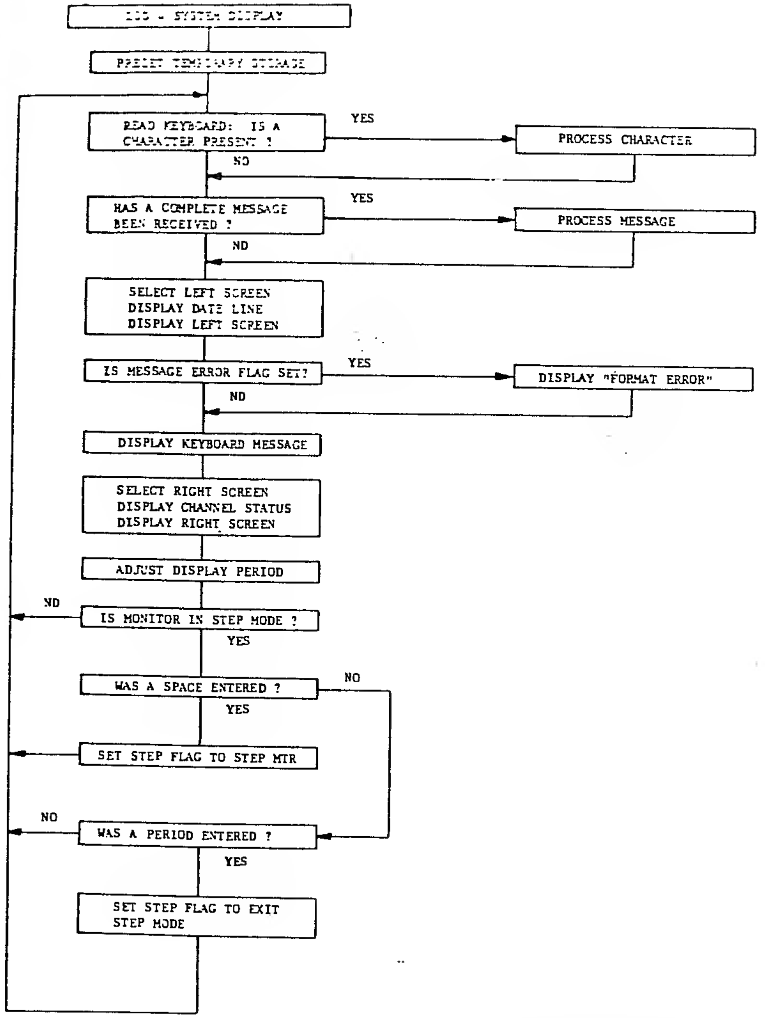
04600	0000	0000	0000	0000	0000
04601	0000	0000	0000	0000	1000
04602	5120	2600	0263	2204	3052
04603	6731	2513	0000	0021	1730
04604	5673	0514	0260	0031	0740
04605	5170	0000	0271	7300	0003
04606	7671	0517	3000	0046	7112
04607	6110	2600	0501	0040	0200

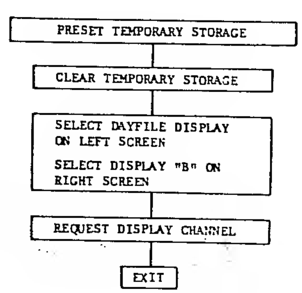
5 GROUPS OF 4 OCTAL DIGITS
(DISPLAYS C, D, AND E)

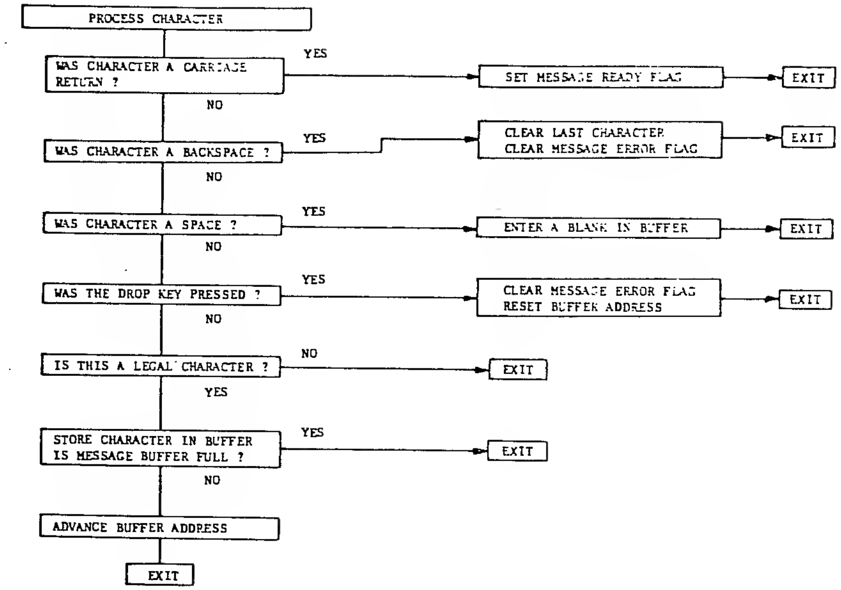
DSD DISPLAYS C THRU G

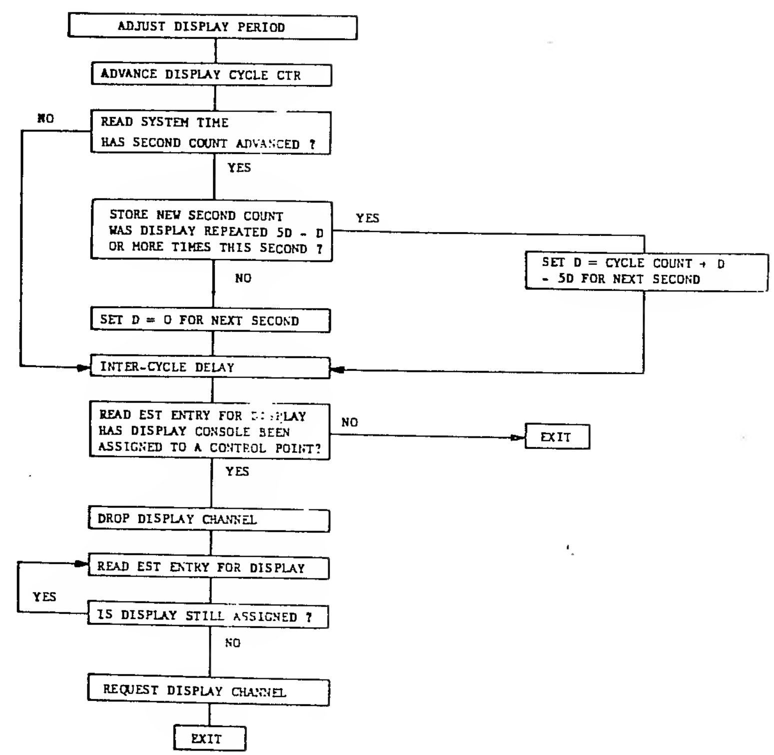
Figure 4

are then read, and the file name and priority displayed. These lists are updated only at intervals of 1/10 of a second in order to reduce unnecessary read pyramid conflicts.


SYSTEM DISPLAY
PRESET TEMPORARY STORAGE
READ KEYBOARD: IS A CHARACTER PRESENT ?
YES
NO
PROCESS CHARACTER
HAS A COMPLETE MESSAGE BEEN RECEIVED ?
YES
NO
PROCESS MESSAGE
SELECT LEFT SCREEN
DISPLAY DATE LINE
DISPLAY LEFT SCREEN
IS MESSAGE ERROR FLAG SET?
YES
NO
DISPLAY "FORMAT ERROR"
DISPLAY KEYBOARD MESSAGE
SELECT RIGHT SCREEN
DISPLAY CHANNEL STATUS
DISPLAY RIGHT SCREEN
ADJUST DISPLAY PERIOD
IS MONITOR IN STEP MODE ?
NO
YES
WAS A SPACE ENTERED ?
NO
YES
SET STEP FLAG TO STEP MTR
WAS A PERIOD ENTERED ?
NO
YES
SET STEP FLAG TO EXIT STEP MODE


PRESET TEMPORARY STORAGE
CLEAR TEMPORARY STORAGE
SELECT DAYFILE DISPLAY ON LEFT SCREEN
SELECT DISPLAY "B" ON RIGHT SCREEN
REQUEST DISPLAY CHANNEL
EXIT


PROCESS CHARACTER
WAS CHARACTER A CARRIAGE RETURN ?
YES
SET MESSAGE READY FLAG
EXIT
NO
WAS CHARACTER A BACKSPACE ?
YES
CLEAR LAST CHARACTER
CLEAR MESSAGE ERROR FLAG
EXIT
NO
WAS CHARACTER A SPACE ?
YES
ENTER A BLANK IN BUFFER
EXIT
NO
WAS THE DROP KEY PRESSED ?
YES
CLEAR MESSAGE ERROR FLAG
RESET BUFFER ADDRESS
EXIT
NO
IS THIS A LEGAL CHARACTER ?
NO
EXIT
YES
STORE CHARACTER IN BUFFER
IS MESSAGE BUFFER FULL ?
YES
EXIT
NO
ADVANCE BUFFER ADDRESS
EXIT


ADJUST DISPLAY PERIOD
ADVANCE DISPLAY CYCLE CTR
READ SYSTEM TIME
HAS SECOND COUNT ADVANCED ?
NO
YES
STORE NEW SECOND COUNT
WAS DISPLAY REPEATED 5D - D OR MORE TIMES THIS SECOND ?
YES
SET D = CYCLE COUNT + D - 5D FOR NEXT SECOND
NO
SET D = 0 FOR NEXT SECOND
INTER-CYCLE DELAY
READ EST ENTRY FOR
HAS DISPLAY CONSOLE BEEN ASSIGNED TO A CONTROL POINT?
NO
EXIT
YES
DROP DISPLAY CHANNEL
READ EST ENTRY FOR DISPLAY
YES
IS DISPLAY STILL ASSIGNED ?
NO
REQUEST DISPLAY CHANNEL
EXIT

CONTROL DATA CORPORATION

Development Division - Applications

THE JOB DISPLAY, DIS

Chippewa Operating System

10/15/65
REV. 1

DIS JOB DISPLAY

INTRODUCTION

DIS is the name of the Chippewa Operating System peripheral program that monitors console - keyboard activity for a job assigned to a particular control point. DIS must be loaded in as many PPU units as the number of control points for which it is required. The package is usually located on the system disk in the peripheral library; therefore, its name will appear in the PLD (Peripheral Library Directory). DIS may be brought to a control point in any one of three ways:

1. Typing "A. DIS CR" when DSD (system display) is active.
2. Inserting a DIS control card.
3. A central memory program requesting DIS through a call to MSG.

DIS is concerned with the following functions:

- Job Displays
- Monitoring Keyboard Activity
- Processing Requests for job control debugging for only the job assigned to its control point.

DIS operates during the time the job to which it is attached has the control point. If it is desired to manually release DIS, a drop request is made, which in turn causes a drop PPUs to be issued by the PPU containing DIS. DIS would have to again be loaded for future use by this control point. If an error condition (error byte becomes non-zero in CP area) the program will drop itself; a check of this nature is made on each iteration through the master control loop.

OPERATIONS

CONSOLE DISPLAY

One of the prime functions of DIS is displaying information concerning the status of the job at the control point to which it is attached. To do this, DIS outputs information in the form of display coded characters (see SIPROS DEF Manual) and necessitates issuing X and Y coordinate

values followed by the string of 6-bit display code characters. The screen of each CRT may be considered a grid of points as follows:

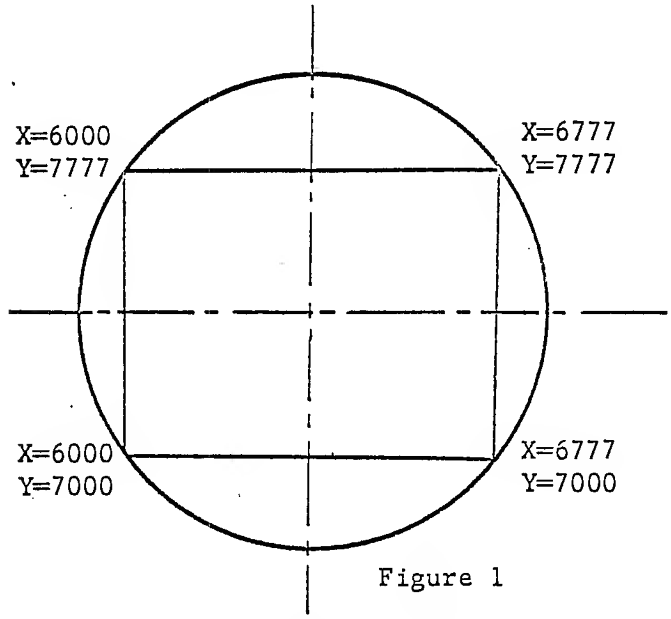


Figure 1

The coordinates (x=6xxx, Y=7xxx) specify the position of the first display coded character follow. Thereafter, the x coordinate is advanced (by one character space) along the increasing x axis but the y coordinate remains constant until another y coordinate (7xxx) is issued. For example, the dayfile display program in DSD uses the area 7200 to 7660 and form 6000 to 6777.

In the dayfile display, the y coordinate is allowed to increase by +1, from 7646 to 7660 on each cycle of the master control loop. As will be seen, this has the effect of "rolling" the display upwards on the screen. The use of the console display is quite simple, involving only the outputing of appropriate x and y coordinates followed by the display coded string of characters.

Besides the formation of display characters and screen positioning, the program also controls the brightness (or intensity) of the image on the screen. The latter increases in proportion to the number of times per second the display coded information is presented to the console. To maintain a stable visable image, the code must be output to the console at least every 1/25 of a second. More repetitions per second will produce a brighter image. A delay loop is commonly employed to control the image output period.

Example of display loop:

```
        IJM   *+3, 10B      .Jump if channel 10B inactive
        DCN        10B      .Disconnect channel 10B
LOOP    FNC   7001B,10B     .Select 32 Char/Line left screen
        ACN        10B      .Activate channel
        LDC   7000B         .A= Y coordinate
        OAN        10B      .Output Y coordinate
        LDC   6337B         .A= X coordinate
        OAC        10B      .Output X coordinate
        LDC        16       .A= No. Words to output
        OAM   Buffer,10B    .Output from buffer
        LDN        01B      .Set A= No. milliseconds delay
        SHN        9        .Convert for LOOP
DELAY   SBN        1        .2 us delay loop
        PJN   Delay
        LJM   Loop          .End of millisecond loop.
```

DISPLAY PLACEMENT

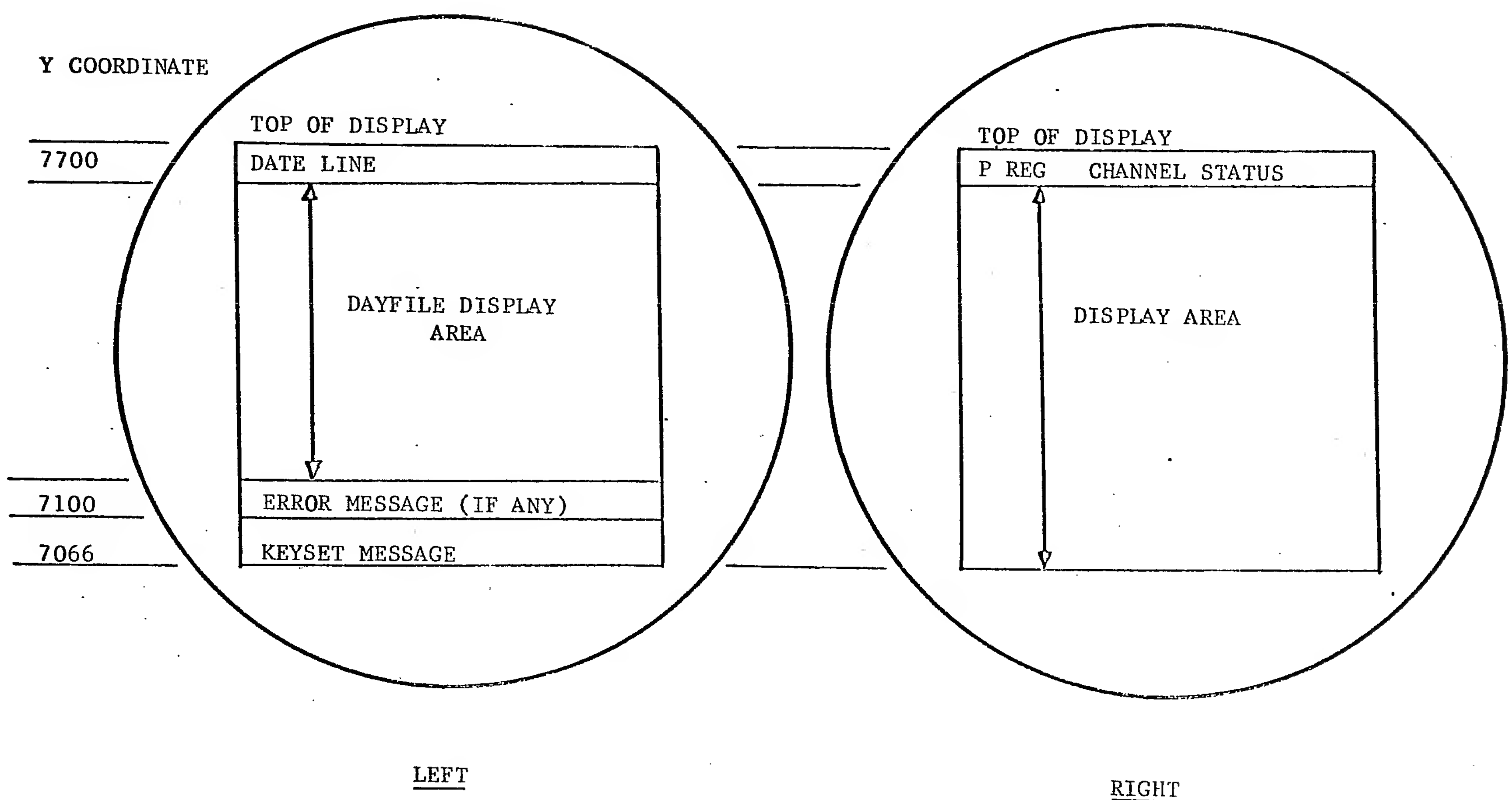


Figure 3

The logic in DIS which controls console output is not basically different from the above example. The overall scheme can be visualized as follows:

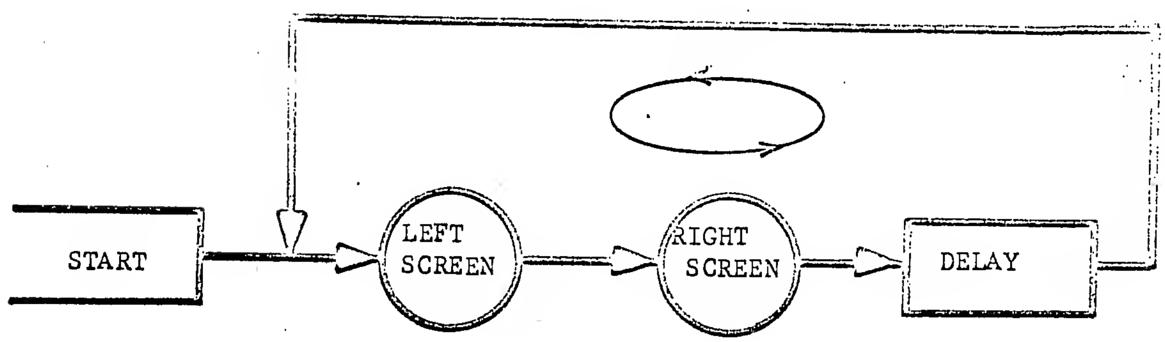


Figure 4

To maintain the delay function, a parameter may be inserted into the ADJUST DISPLAY PERIOD and PRESET INITIAL VALUES ROUTINES. It has the form: LDN nn, where nn is the number of milliseconds (from 00 to 47_8) in the delay. For maximum brightness, this is preset to nn=00; it can be modified, however. Since other operations (keyboard monitoring) take place in DIS, this constant also effects the total sensitivity of the whole system. For normal operations, it is not necessary to alter this constant. Notice in figure 4 the image is established through continuous trips around the display loop.

<u>KEYBOARD</u>

Along with the display screens at the console unit is a typewrite-like keyboard containing keys for alphabetic, numeric and special characters. These, in display code, are listed on page 46 of the manual on CODES for the 6000 Computer System. Keys are also present for the following special purposes:

KEY	DISPLAY CODE
Carriage Return (CR)	60_8
Backspace	61_8
Space	62_8

These keys are used by the KEYBOARD MONITORING routine to control the proper filling of the KB (keyboard) assembly buffer (location 1300/1377 in DIS). A CR is interpreted by DIS as an end of message. The backspace key will erase the last character input. The drop key (code 55) will cause the string of characters of the current message, input to that time, to be cleared; the next character keyed will be treated as the first of a new message.

When it is desired to interrogate the keyboard for keyed information, an input to A is given. If there was a character keyed, the console unit controller will return the character, in display code, in the lower 6 bits of a 12-bit byte (the high order 6 bits are cleared to 00). If no key was activated, however, the controller will return an all zero 12-bit byte.

On each cycle of the Master Control Loop, DIS examines the data at the keyboard. If a valid character is input, the byte is stored away in a contiguous manner in the KB assembly buffer. A cleared byte (0000) will cause no data to be stored. Appropriate action is taken when the CR, backspace or drop key is activated. When a CR is recognized, a flag, called the keyset Ready flag is set to 0001; representing the "on" condition. This flag is examined once during each control cycle. If not set (0000) no action is taken; if set (0001), however, the message in the KB Assembly buffer is interpreted, processed and the flag reset to 0000.

Figure 5 illustrates the position of the keyboard processing in the control sequence of DIS (see figure 4 also):

FUNCTIONAL SEQUENCE OF DIS

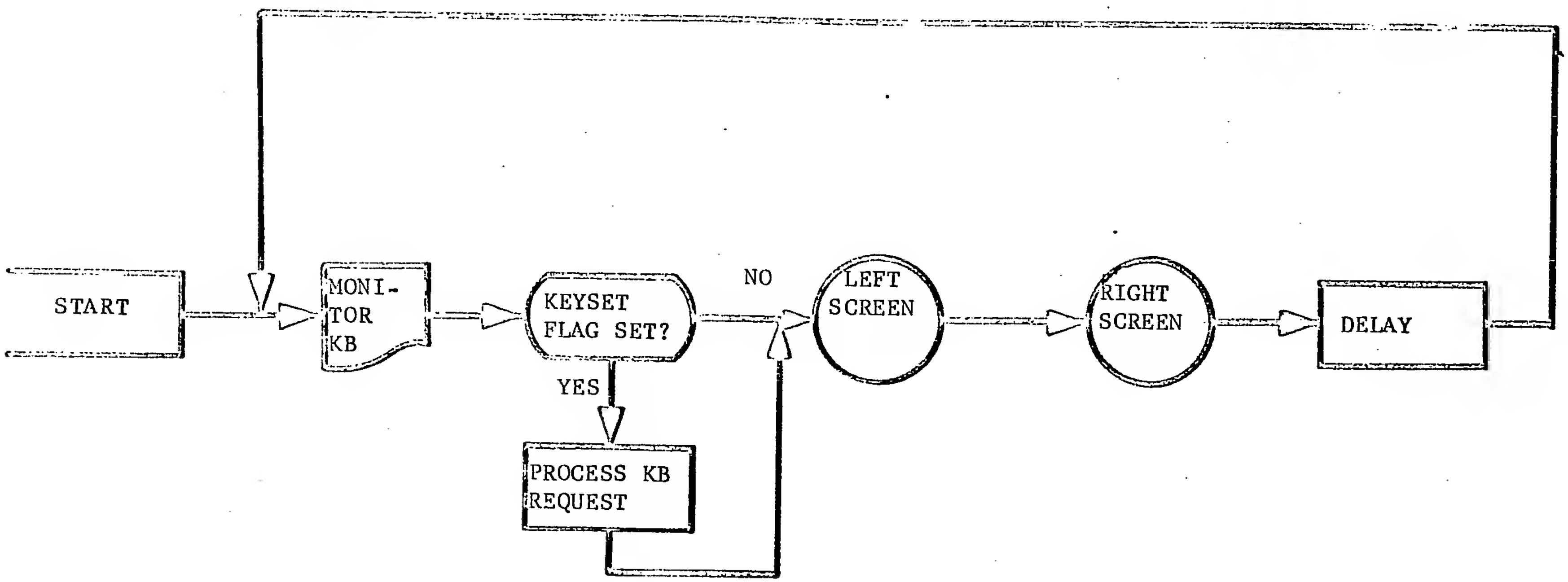


Figure 5

MASTER CONTROL PROGRAM

Figure 5 combines several of the functions which make up the MCP (MASTER CONTROL PROGRAM):

- Keyboard Monitoring
- Request Processing initiation
- Left Screen Display
- Right Screen Display
- Display Period Adjustment

Only a few more functions need to be added to produce the operations performed by the MCP. Regardless of the display mode selected (A, B, ..., G) there are permanent displays on both the left and right screens. On the left appears the time and date line stored in Central Memory locations 30/37; this is displayed at Y coordinate 7700. At Y coordinates 7100 and 7066 are the error message (if any) and the contents of the KB Assembly Buffer (at that point in time), respectively. The current Central Processor P register and the status of all 12 data channels are displayed at Y coordinate 7700 on the right screen. In the functional diagram (figure 5) these displays may be thought of as belonging to the left and right screen displays.

One function remains: breakpoint monitoring. The user is given the opportunity to request (via the keyboard) a breakpoint debugging action. This routine reads the CP P register and if the contents match the requested breakpoint address (in P 52/53) the Central Processor is dropped from the control point and the word at the breakpoint (saved in P40/144 during breakpoint initiation) is restored in CM. This function follows the display period adjustment routine. All functions of the MCP are illustrated in detail in MCP flow chart (see appendix A).

DISPLAY PROGRAMS

DIS is able to provide four different types (modes) of displays. The operations indicated in the two circles in Figure 5 refer to the particular display and its associated display program selected for that screen for discussion purposes. The permanent displays were considered here as well. Initially, DIS will put up the dayfile display on the left screen and the job status infor on the right. These can be altered, if desired, by keying the following request.

LR. "CR"

The mode code placed in the L position will bring the display for that code to the left screen and one placed in R will specify the right screen display. The address for the left and right screen programs are stored in locations 70 and 71, respectively.

DISPLAY MODES

CODE	DISPLAY INFORMATION
A	Day File
B	Job Status
C, D, E	Program Storage
F, G	Data Storage

Day File Display - MODE A

This program displays the contents of the DFB (Day File Buffer) between Y coordinates 7660 (top) to 7200 (bottom). This coordinate is stored in location 64. The most current being displayed message will always appear at the bottom of the display. Up to 3/10 messages can appear between these limits. After each message is displayed, the Y coordinate is decremented by 12_8.

A pointer in location 65 is maintained to indicate the address of the DFB message to be displayed at the top of the display. Initially this pointer equals the "OUT" address of the DFB status word (CM location 0003); this is

advanced as the number messages between the current address (at 65) and the "INPUT" address exceeds 31_{10}. When 31 or fewer DFB messages are between (65) and "INPUT", the display stays constant with new messages being displayed at the bottom of the display as they appear in the DFB.

If, however, the number of messages in the DFB exceeds the maximum number that can be processed at one time, the display will not remain fixed. Rather, it will "roll up" on the face of the screen. This gives the impression that when the message now at the head of the display reaches the top it is rolled off the screen and a new (and more current) message enters the display at the bottom. This will continue until the number of DFB messages between the address stored in 65 and the INPUT DFB status indicator becomes less than 31. This rolling is accomplished by allowing the beginning Y coordinate to vary from 7646 to 7660 (in increments of 1) on each cycle of the DIS master control loop. Therefore, 12_8 iterations will be necessary to roll off a message. As each message is rolled off the screen, the DFB pointer in 65 is advanced to the address of the next message in the DFB. All display references are made relative to this address. Eg: if 65 contained 2334 then the first (top most) message would be picked up from the CM location 002334. Then the next 30 messages (not the next 30 cells) to be displayed will be picked up. When the message beginning at CM 002334 is rolled off the display, the address in 65 will be set to point to the first word of the next DFB message. If the message at CM 002334 is three cells long, 65 will be set equal to 002337. Whenever a message is rolled off, the Y coordinate is reset equal to 7646. See the flow chart in Appendix A, A-39 for a detailed description of the process.

Job Status Display - MODE B

This display exhibits the control point status. (W, X, A, ...G, -), last dayfile message the next control state-

ment to be processed and the exchange jump package. This information is gathered from the control point area.

Storage Displays - MODES C, D, E, F, G

Modes C, D, E are primarily used to display program text residing within Central Memory. These form display octal digits in the form of 4 groups of 5 digits each.

Modes F, G, in contrast, display octal digits in 5 groups of 4 digits. These coorespond to the 12-bit PPU words and hence modes F, G are used for data storage.

Besides the above differences, the five modes all share these characteristics. All displays have four fields. A field is the display of the eight words XXXXX0 - XXXXX7. The particular field specification is given by the typed statement:

Xn, m. "CR"

where X = C, D, E, F, G

n = 0 Field 0 begins with m
1 Field 1 begins with m
2 Field 2 begins with m
3 Field 3 begins with m
4 Four consecutive field beginning with m.

m = Central memory relative (to RA) address. This should be of the form XXXXX0. In any case, the low order digits is made 0 if any other digit is specified.

Eg: 1. C2,330. "CR" would set field 2 beginning address equal to 000330 and display 000330 - 000337.

Eg: 2. E3,351. "CR" would set field 3 equal to 000350 and display 000350 - 000357.

All displays give the relative CM address to the left of each entry of the display.

REQUEST PROCESSING

On each iteration through the Master Control Loop the Keyset Ready flag is examined. If the flag has the value

0000 (i.e., not set) the remaining portion of the loop is traversed. If, however, the flag is set, control is given to the INTERPRET KEYBOARD MESSAGE routine. This routine scans the information in the KB buffer and gives control to the proper routine to process the keyed request.

The requests may be classified into the following groups:

1. Display mode selection and mode field specification
2. Central Memory modification
3. Exchange Jump Package and Control Point modification
4. Job control
5. Debugging Aids

The KB interpretive routine first checks for a special format (see list of possible requests following this discussion) and if request is of this gives control to the routine specified (see flow charts A-17 to A-36). If it is not one of these, the statement is examined for a display entry (mode or field change); if it is of this type, control goes to proper display processor. If the statement still can not be identified, it is treated as a possible PP call and the RPL and PLD are reached. If a match is made, a request is set up in 10/14 and a return is made to PP Resident to inform EXEC that there is a request to process. If the request is not a PP call, it is considered to be an error and the message "FORMAT ERROR" is desplayed on the left screen. The Keyset Ready flag is then cleared to 0000 and control is returned to the Master Control Program. For a complete description of individual request processing, consult the DIS flow charts in Appendix A.

DIS REQUESTS

The following commands to DIS refer to the control point to which it is attached. Some of the entries cause the job to be switched away from the CPU (e.g. when the job's exchange package has to be changed). Execution can be resumed using RCP or BKP, Numbers are in octal.

. ENP, 12345.	Set P = 12345. (Next instruction address in exchange package).
. ENA3, 665000.	Set A3=665000 in exchange package.
. ENB2, 44.	Set B2=44 in exchange package.
. ENX5, 2223 4000-0000 0000 0200. (Spacing unimportant)	Set X5=22234000000000000200 in exchange package.
. ENEM, 7.	Set Exit Mode = 7 in exchange package.
. ENFL, 10000.	Set FL=100000 en exchange package. (Storage moved if necessary).
. ENTL, 200.	Set CPU Time Limit ≈ 200_8 seconds.
. ENPR, 5.	Set job Priority = 5.
. DCP	Drop central processor and display exchange package (in display B). Using DIS, the exchange package is displayed in any case if the job does not have status A, B, etc.
. RCP.	Request central processor. This puts the job in W status, and it will take the CPU if its priority is sufficient. The register settings of the exchange package will be used.
. BKP, 44300.	Breakpoint to address 44300 in the program. CPU execution begins at the current value of P and stops when P = 44300. DIS effects this by clearing 44300 to stop the program at that point, and restores the original word when the stop occurs.

. RNS.	Read next control statement and obey it. (During use of DIS the normal advance of control statements is inhibited).
. RSS.	Read next control statement and begin execution. This is like RNS, except that a central program is only brought to central memory, and not executed.
. ENS.xxxxxxxxxxxxxx.	This command allows the entry of any control statement as if it had been entered on a control card. The statement can then be processed using RNS or RSS.
. GO.	This command restarts a program which has paused.
. ONSW3.	Set sense switch 3 for the job.
. OFFSW4.	Switch off sense switch 4 for the job.
. HOLD.	This entry causes DIS to relinquish its display console, but the job is held at its present status. A console must be reassigned to continue use of DIS.
. DROP.	This causes DIS to be dropped and normal execution of the job is continued. It does not mean 'Drop the job.'
. DMP (200, 300).	Dump storage from 200 to 277 in the output file.
. DMP (400).	Dump storage from the job's reference address to 377.
. DMP.	Dump exchange package to output file.

(DMP formats are the same as if used on control cards).

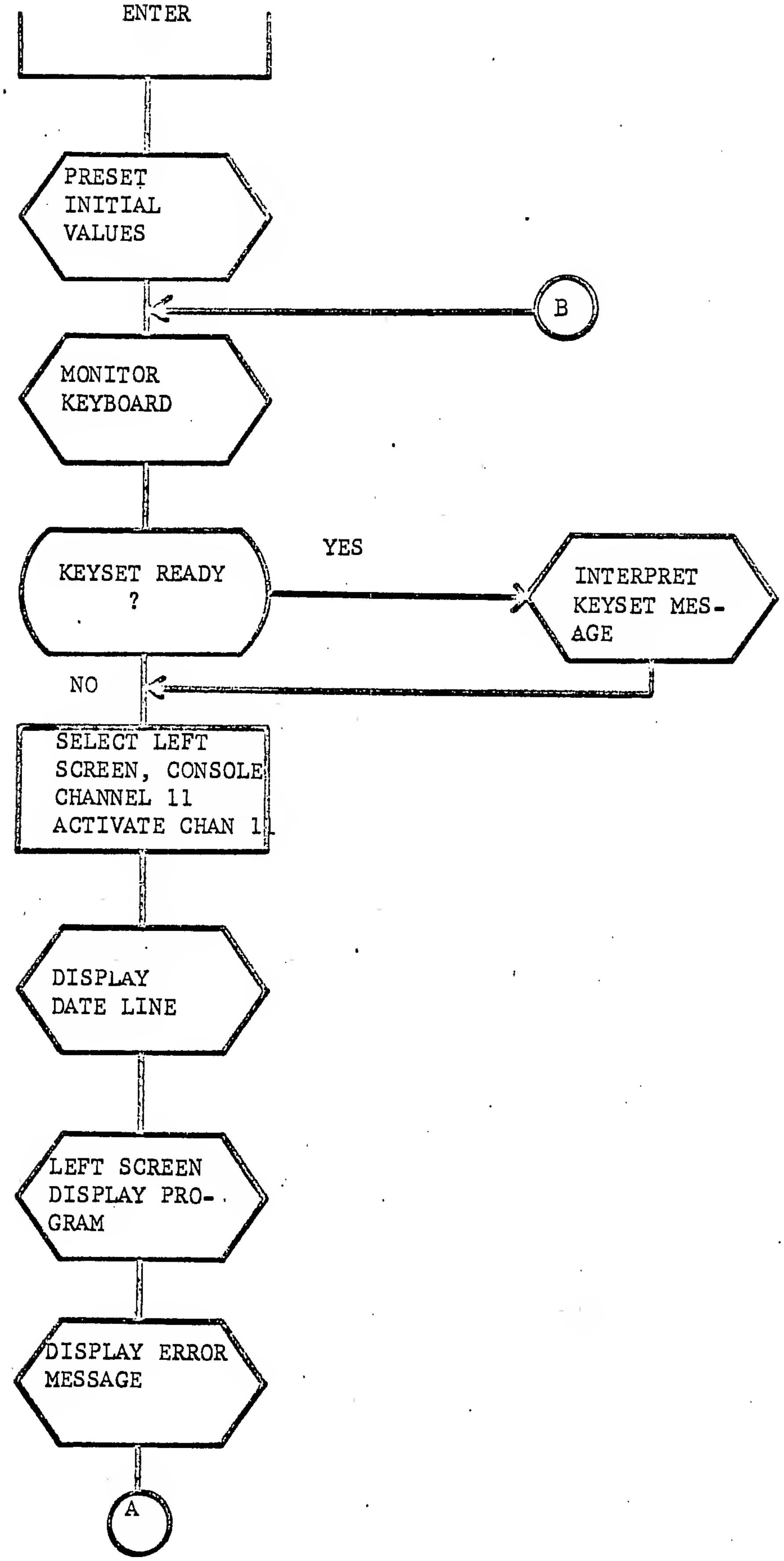
APPENDIX A - DIS FLOW CHARTS

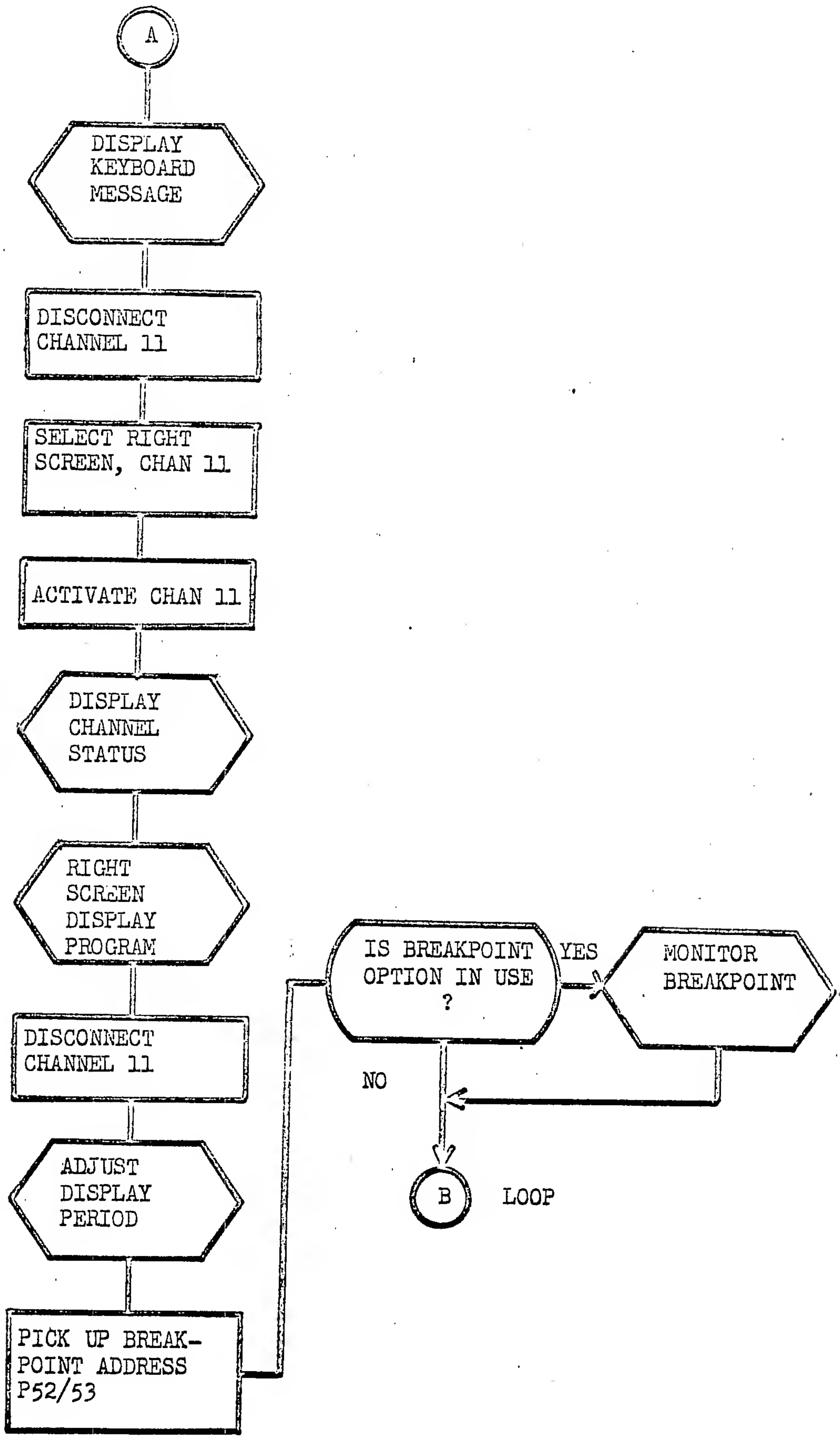
INDEX

INDEX

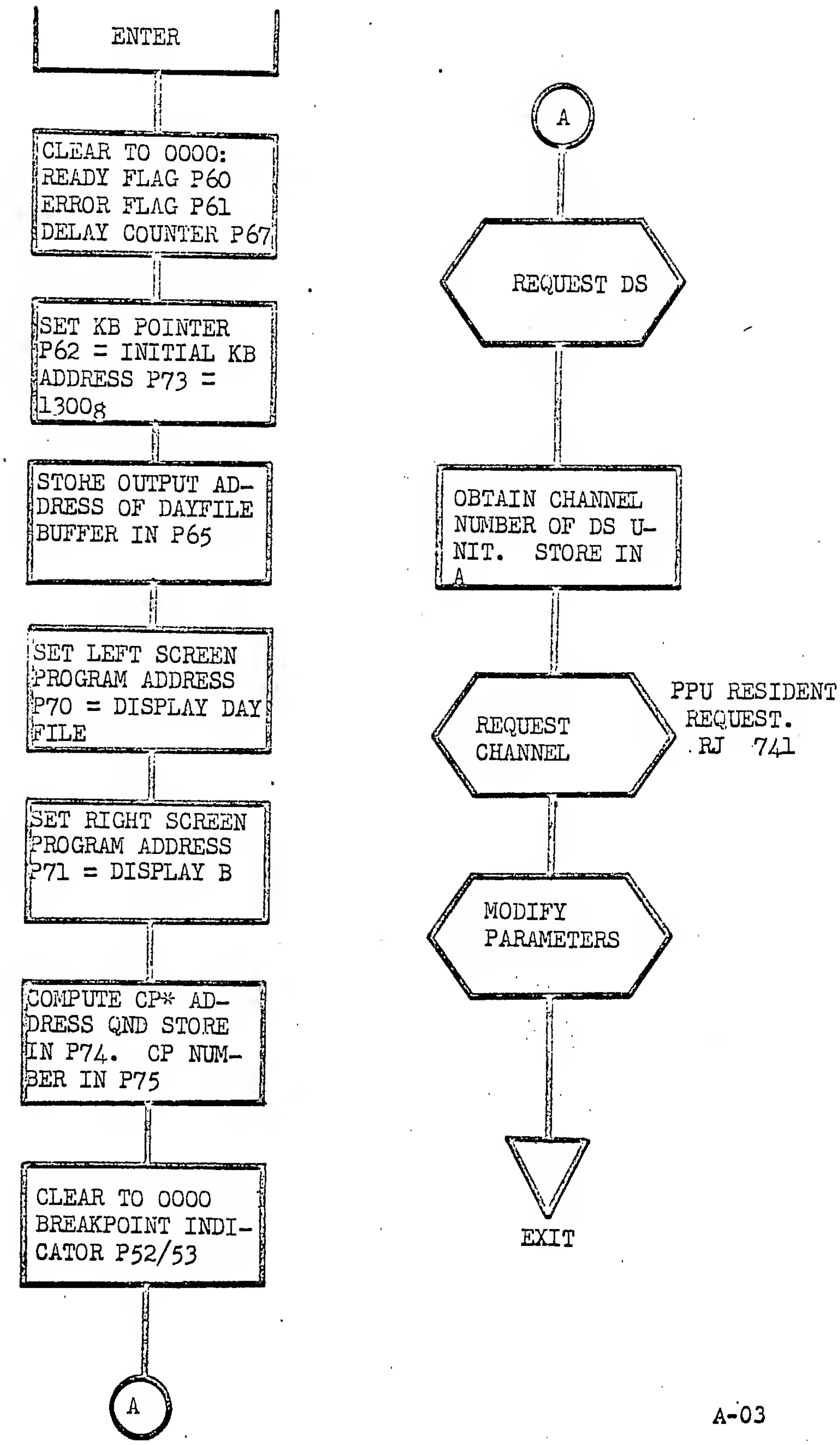
ODIS	TEMPORARY STORAGE ALLOCATION
40/44	BREAKPOINT WORD
50	REFERENCE ADDRESS
51	FIELD LENGTH
52/53	BREAKPOINT ADDRESS
60	KEYBOARD READY FLAG
61	KEYBOARD ERROR FLAG
62	KEYBOARD ADDRESS
63	EQUIPMENT ADDRESS
64	DAYFILE DISPLAY COORDINATE
65	DAYFILE DISPLAY ADDRESS
66	DISPLAY CYCLE COUNTER
67	DELAY COUNT
70	LEFT SCREEN PROGRAM
71	RIGHT SCREEN PROGRAM
73	KEYSET INITIAL ADDRESS
74	CONTROL POINT ADDRESS
75	INPUT REGISTER
76	OUTPUT REGISTER
77	MESSAGE BUFFER

DIS - MAIN CONTROL PROGRAM



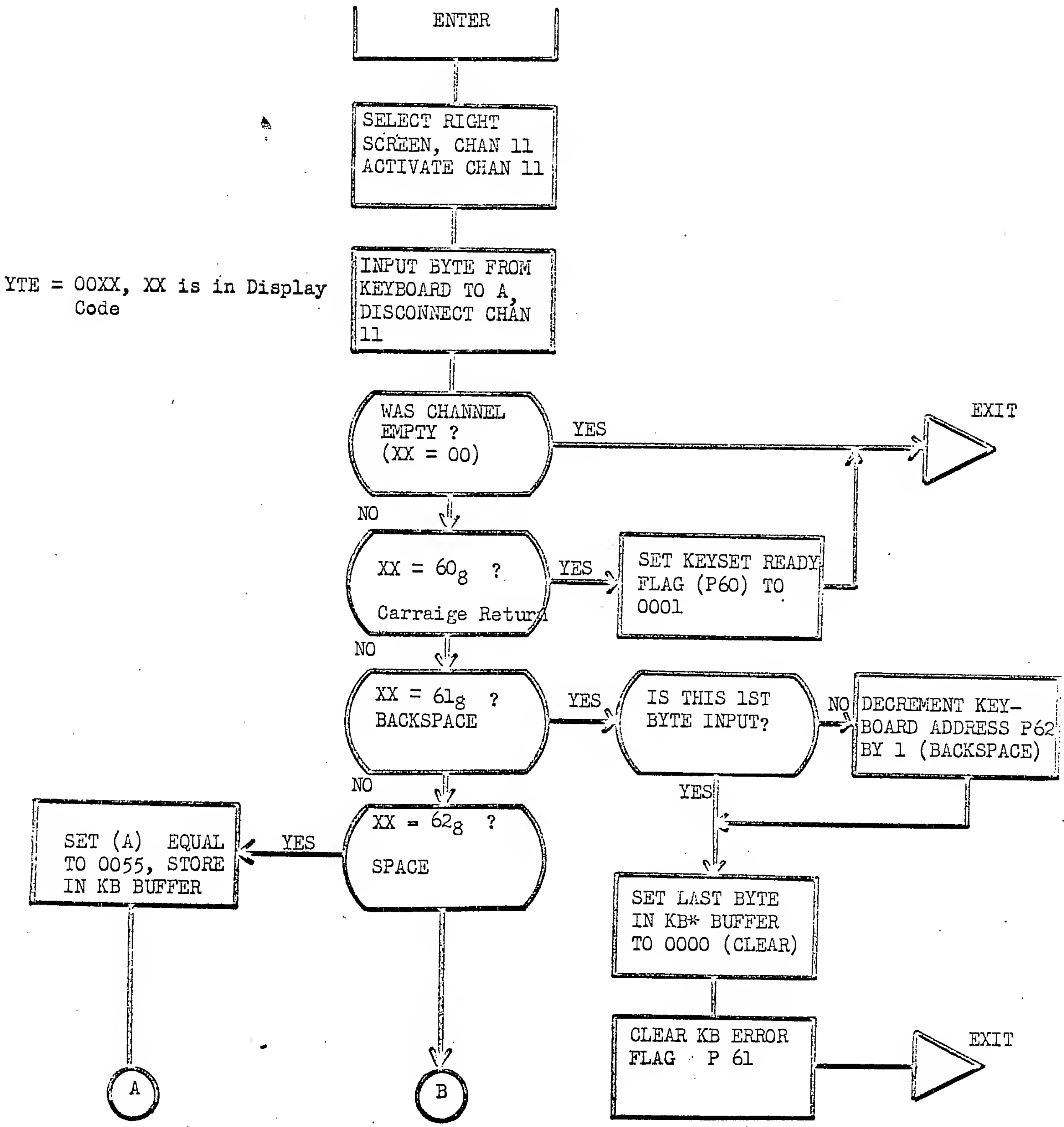

A
DISPLAY KEYBOARD MESSAGE
DISCONNECT CHANNEL 11
SELECT RIGHT SCREEN, CHAN 11
ACTIVATE CHAN 11
DISPLAY CHANNEL STATUS
RIGHT SCREEN DISPLAY PROGRAM
DISCONNECT CHANNEL 11
ADJUST DISPLAY PERIOD
PICK UP BREAK-POINT ADDRESS P52/53
IS BREAKPOINT OPTION IN USE ?
YES
MONITOR BREAKPOINT
NO
B
LOOP

DIS - PRESET INITIAL VALUES



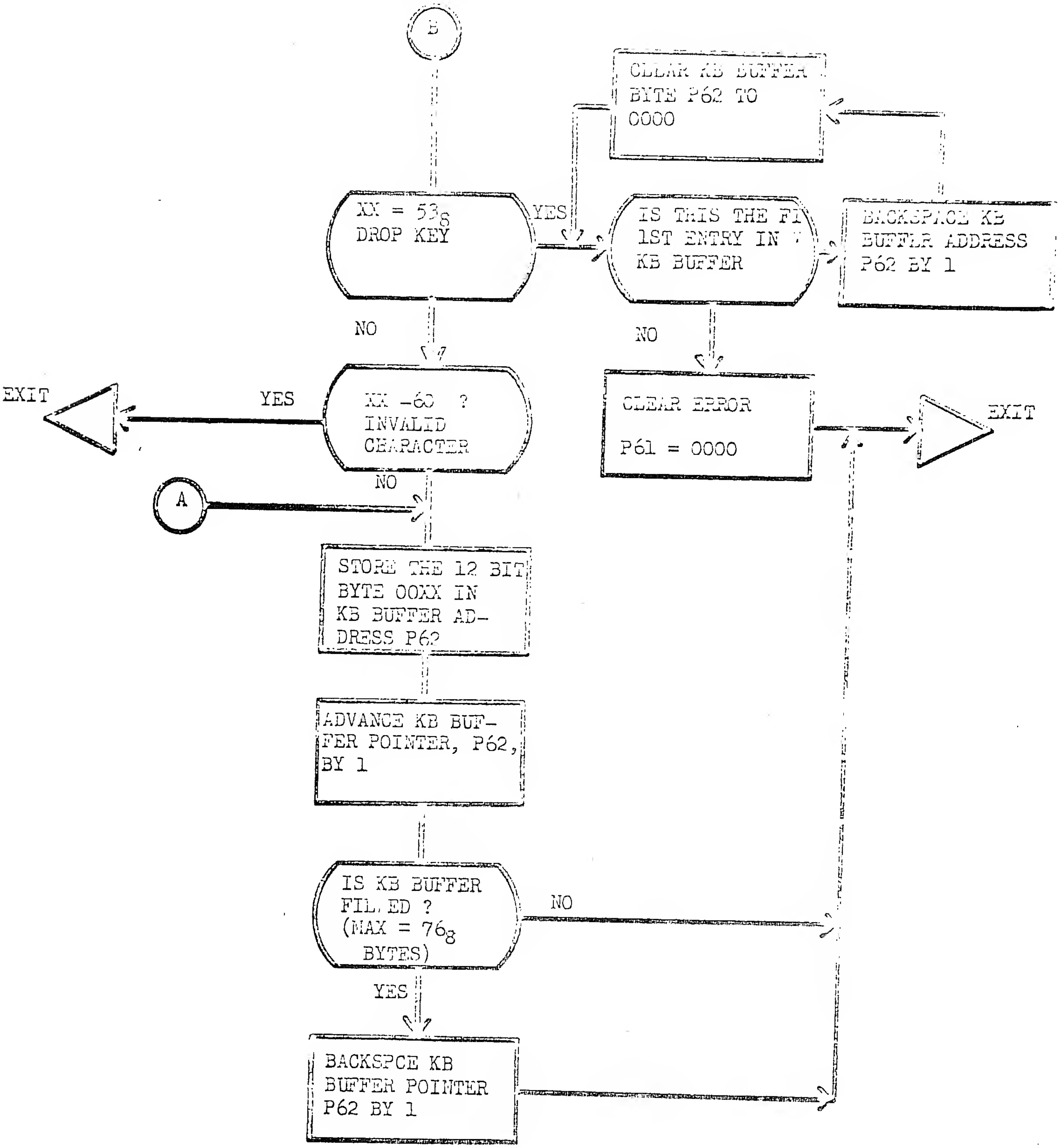
* CP Means Control Point

DIS - MONITOR KEYBOARD

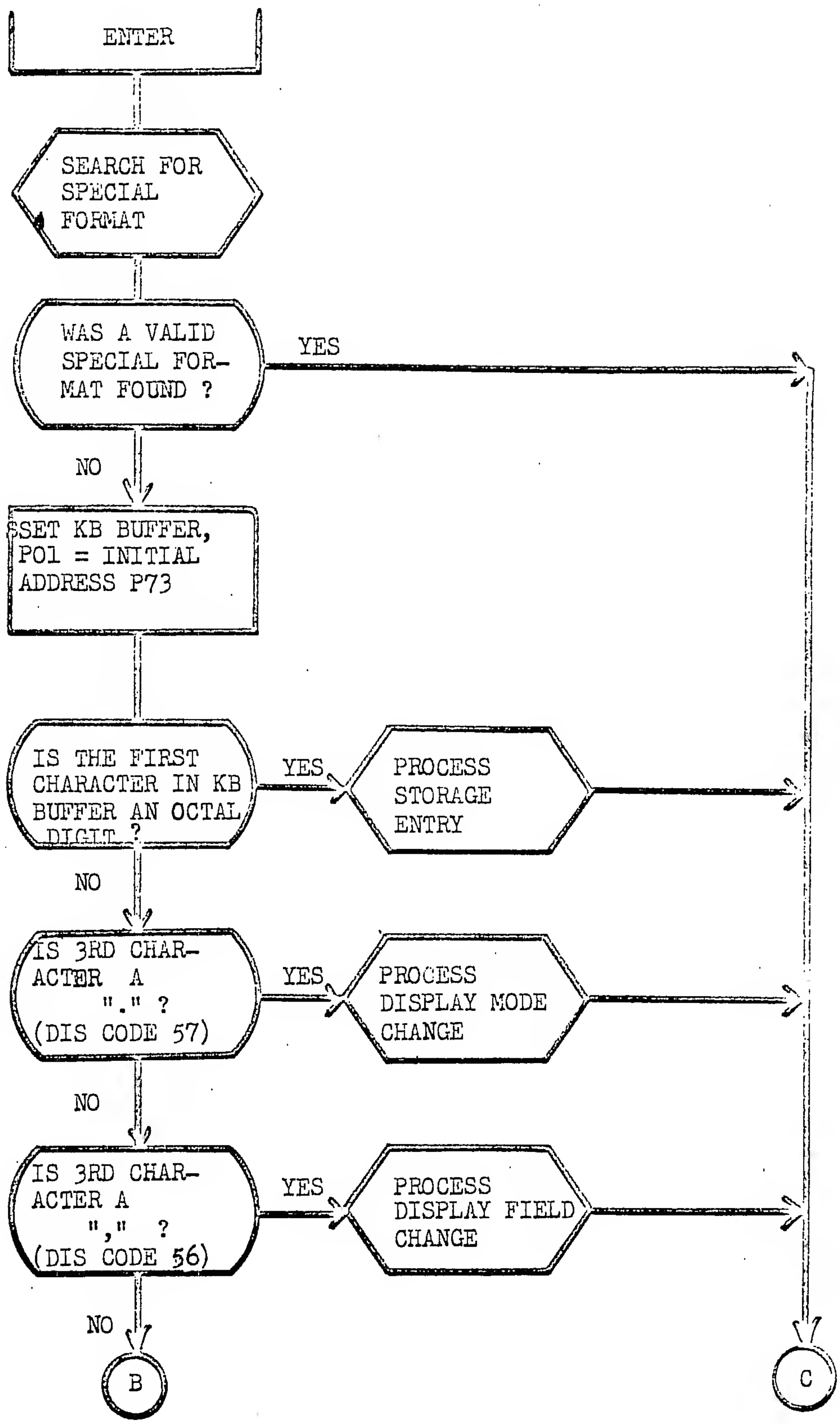


* - KB Means Key Board

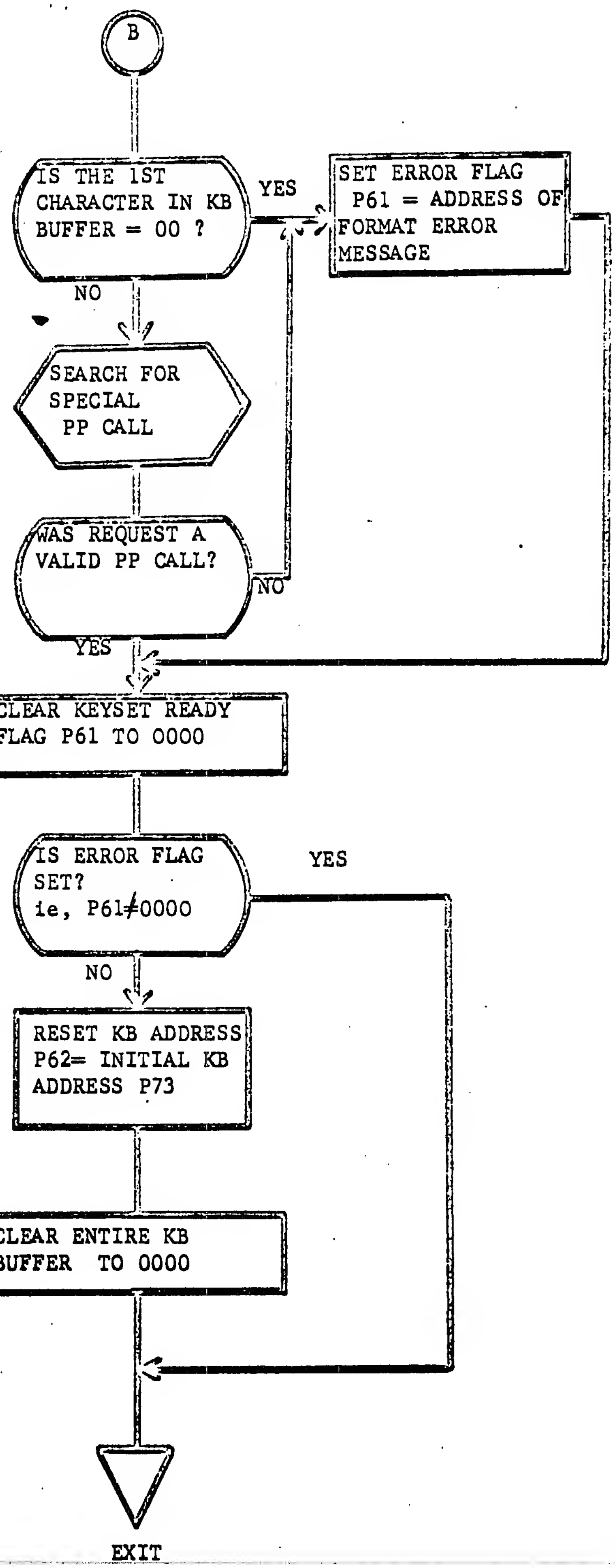
DIS - MONITOR KEYBOARD (CONTINUED)



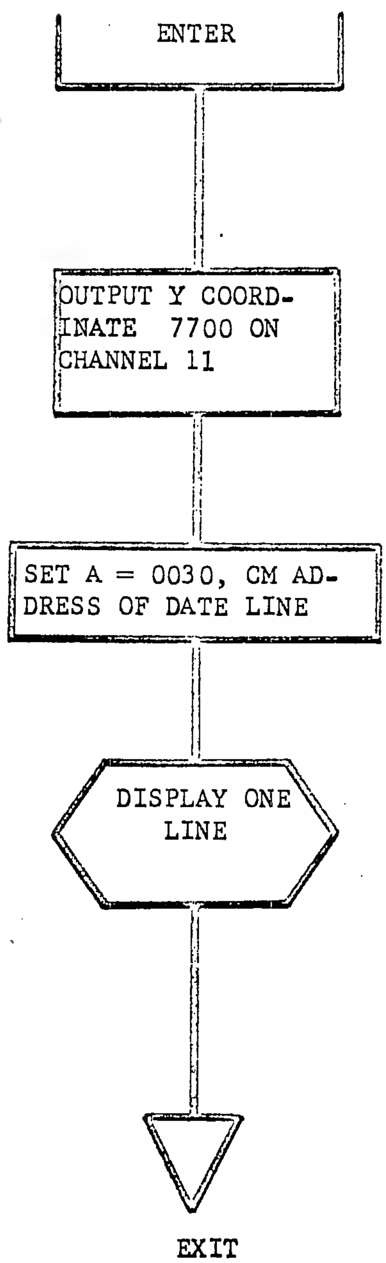
DIS - INTERPRET KEYSET MESSAGE



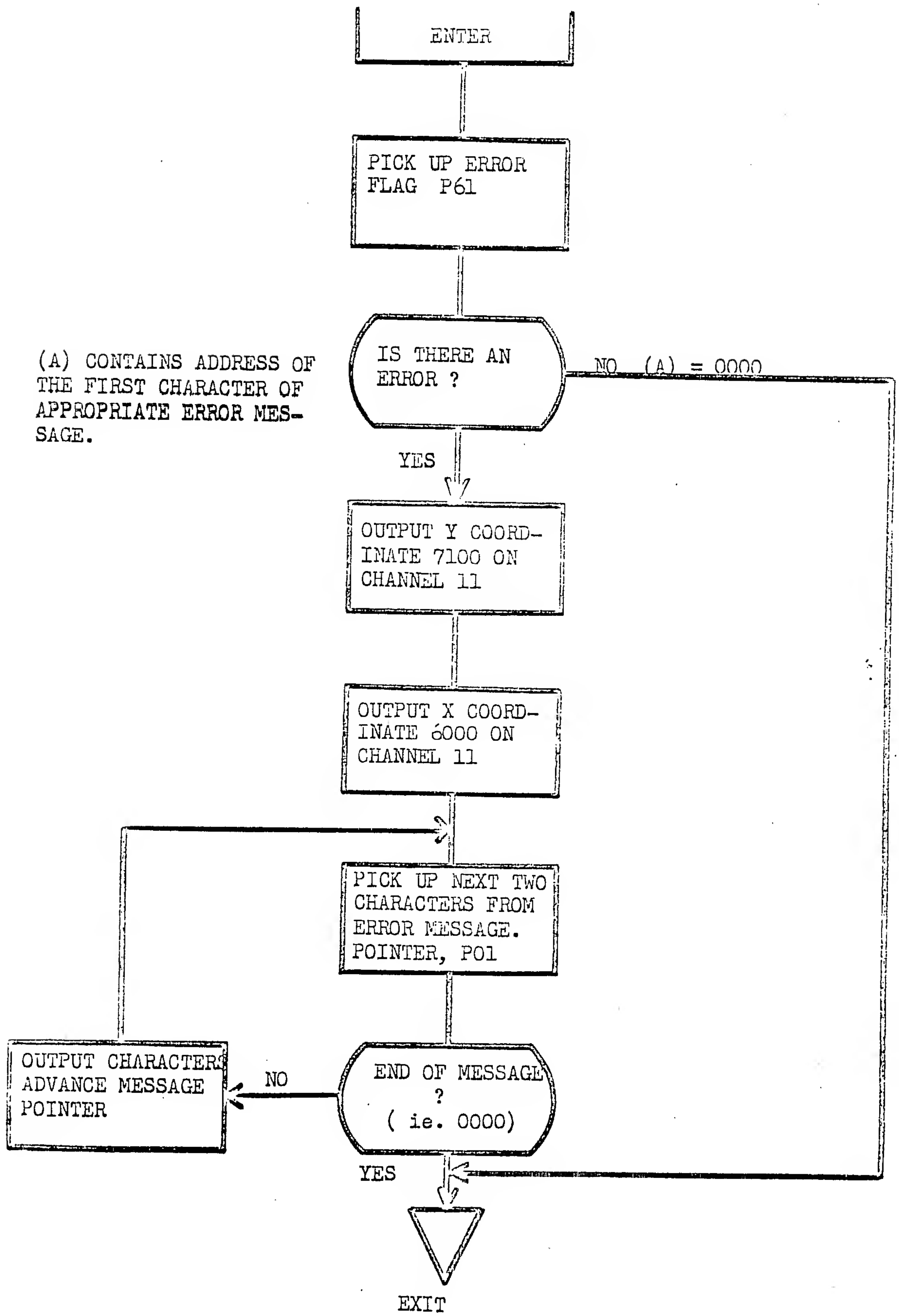
DIS - INTERPRET KEYSET MESSAGE (CONTINUED)



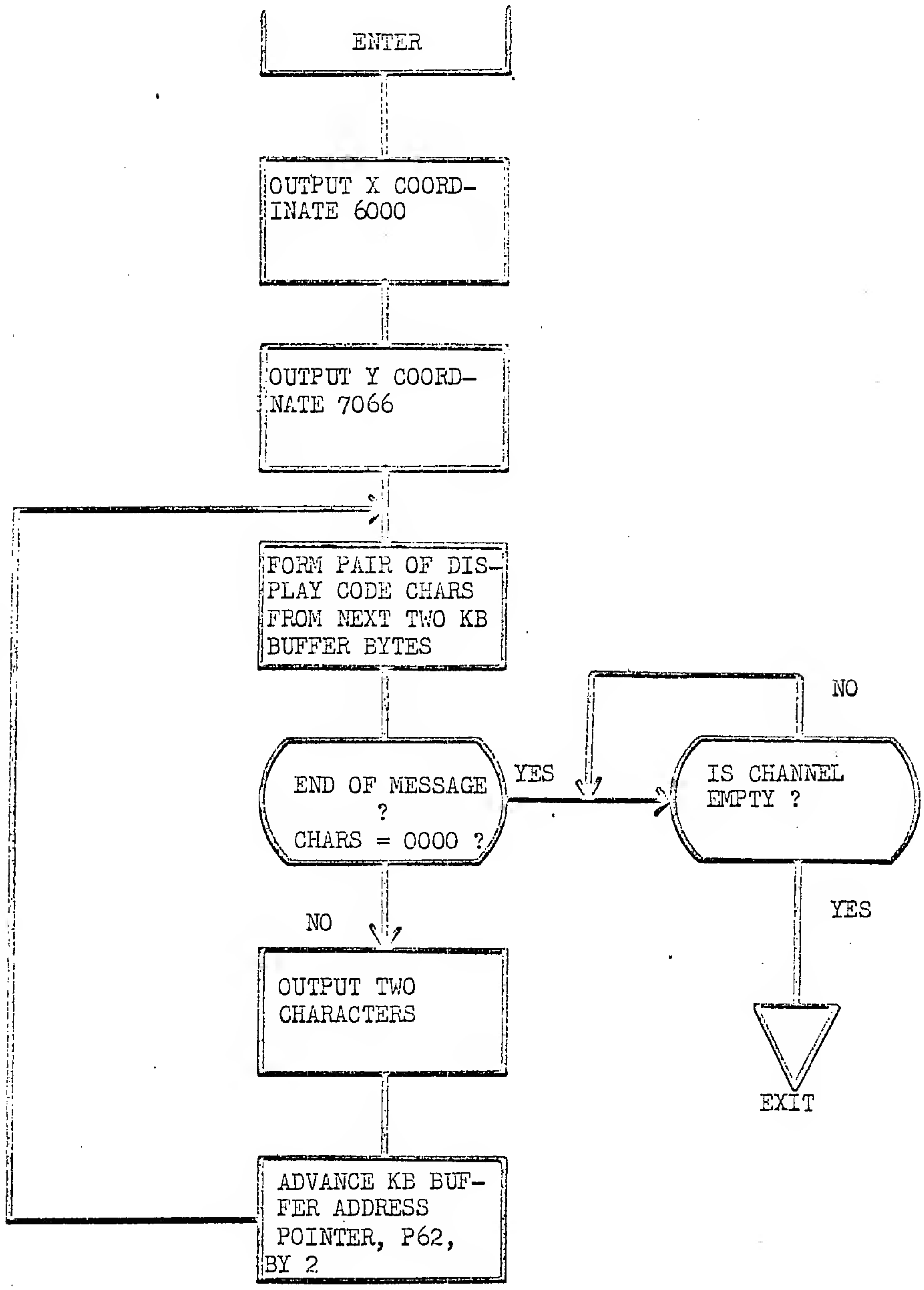
DIS - DISPLAY DATE LINE



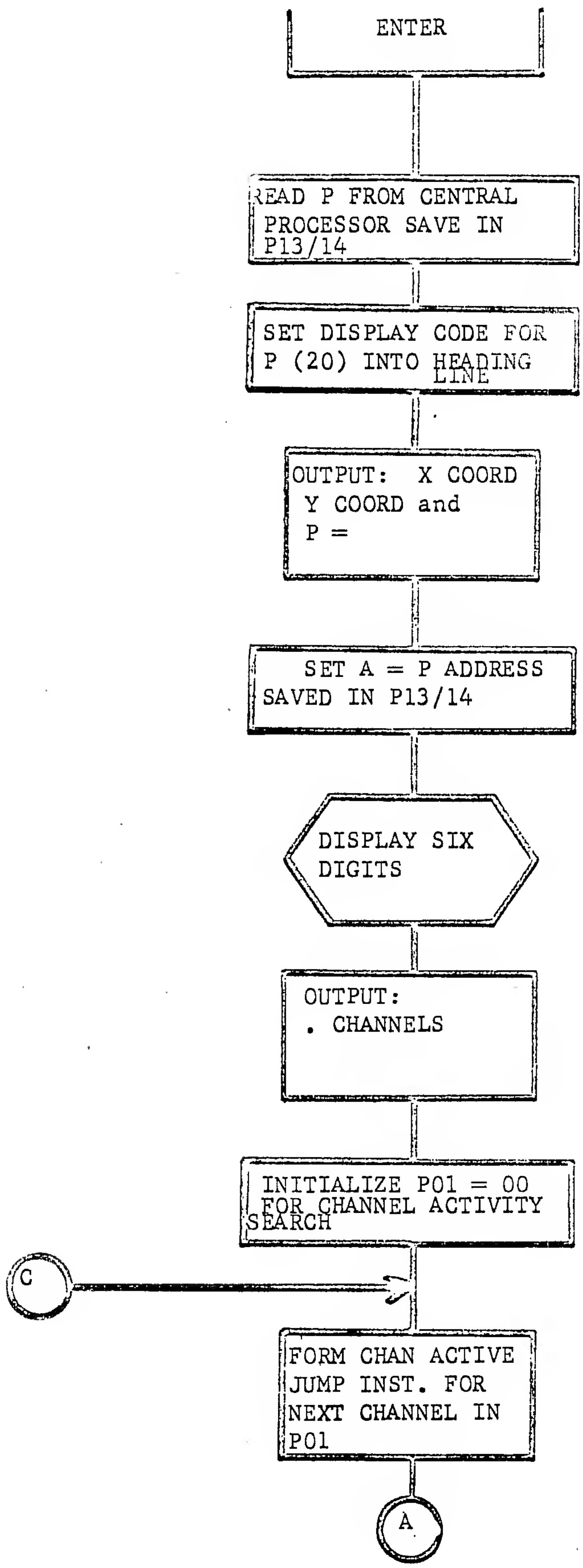
DIS - DISPLAY ERROR MESSAGE



DIS - DISPLAY KEYSET MESSAGE

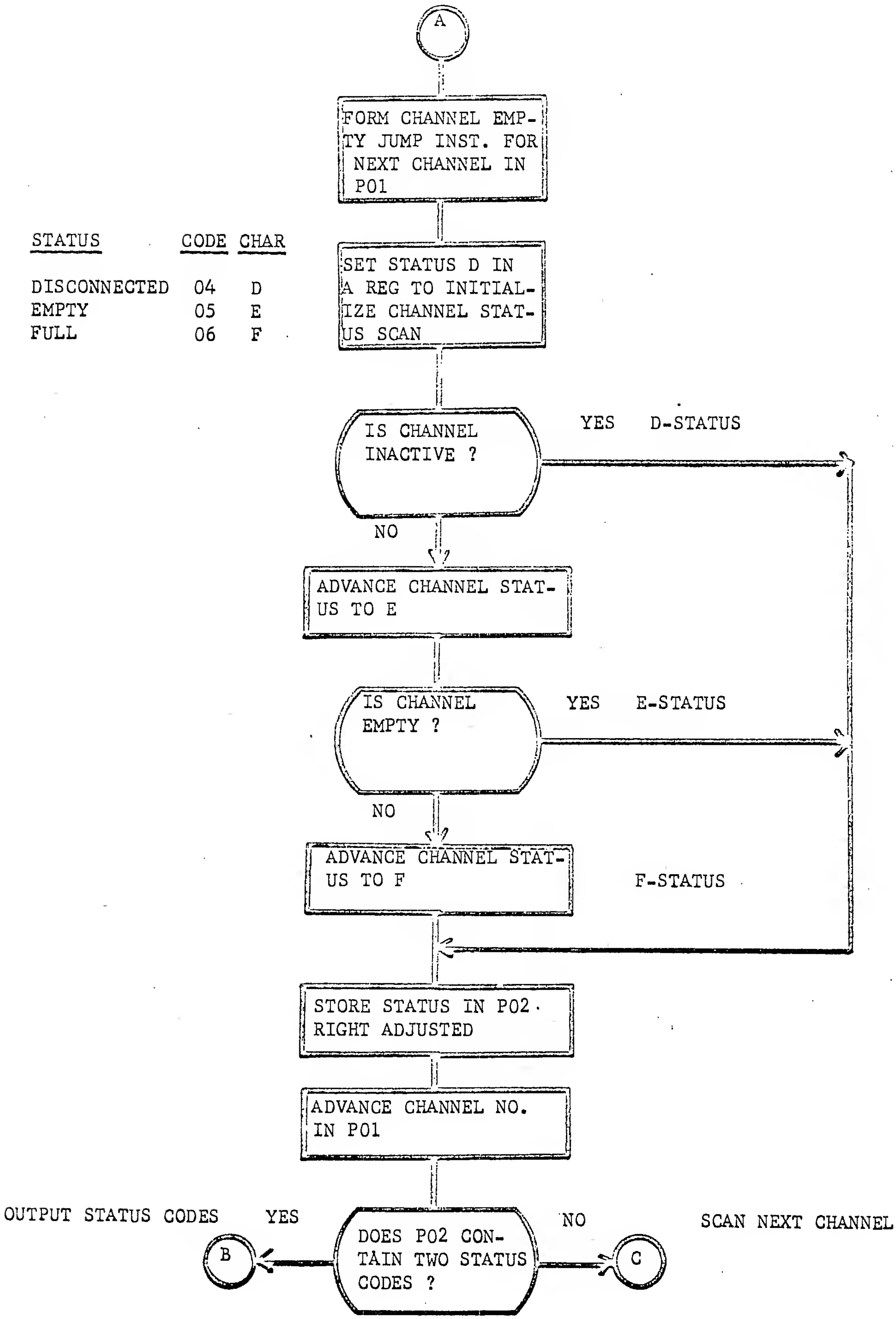


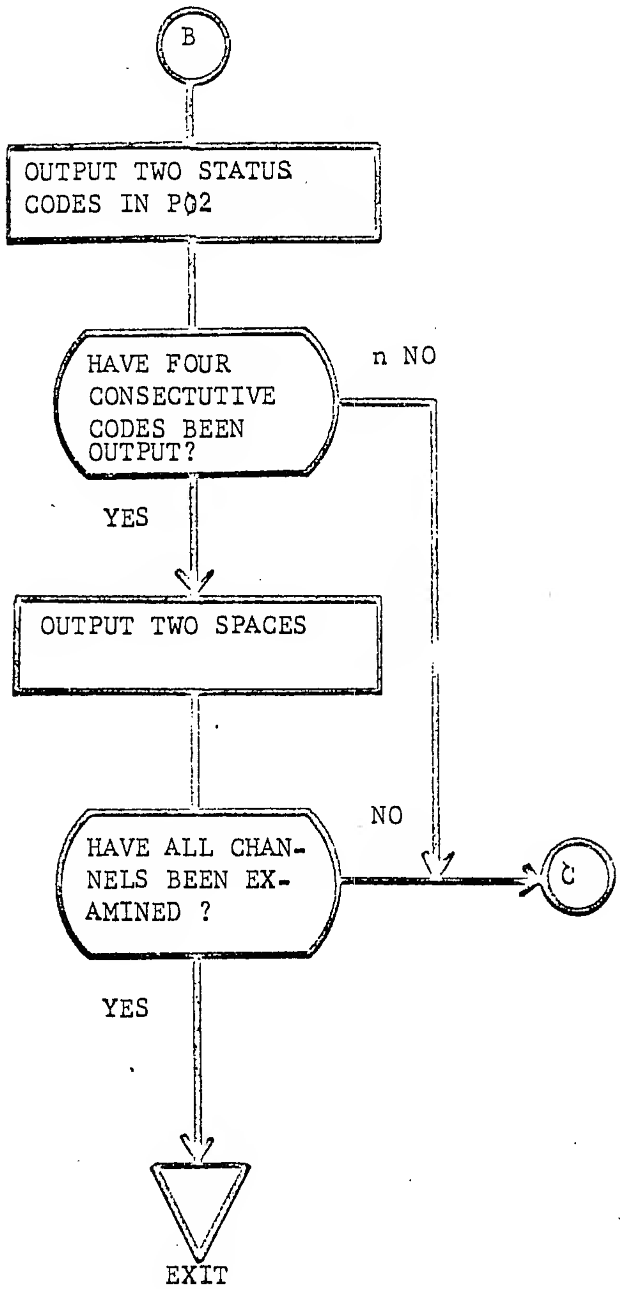
DIS - DISPLAY CHANNEL STATUS



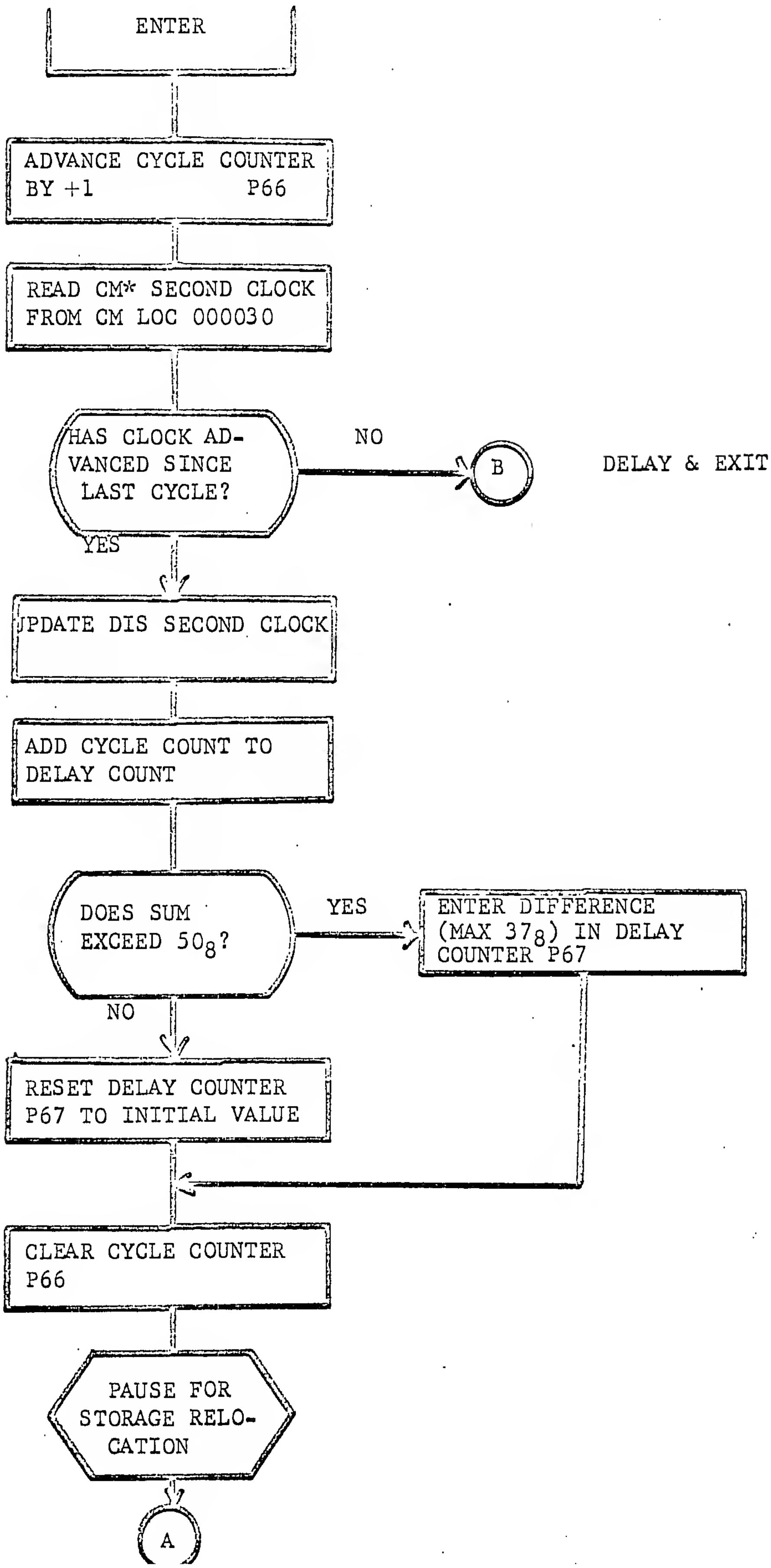
DIS - DISPLAY CHANNEL STATUS (CONTINUED)

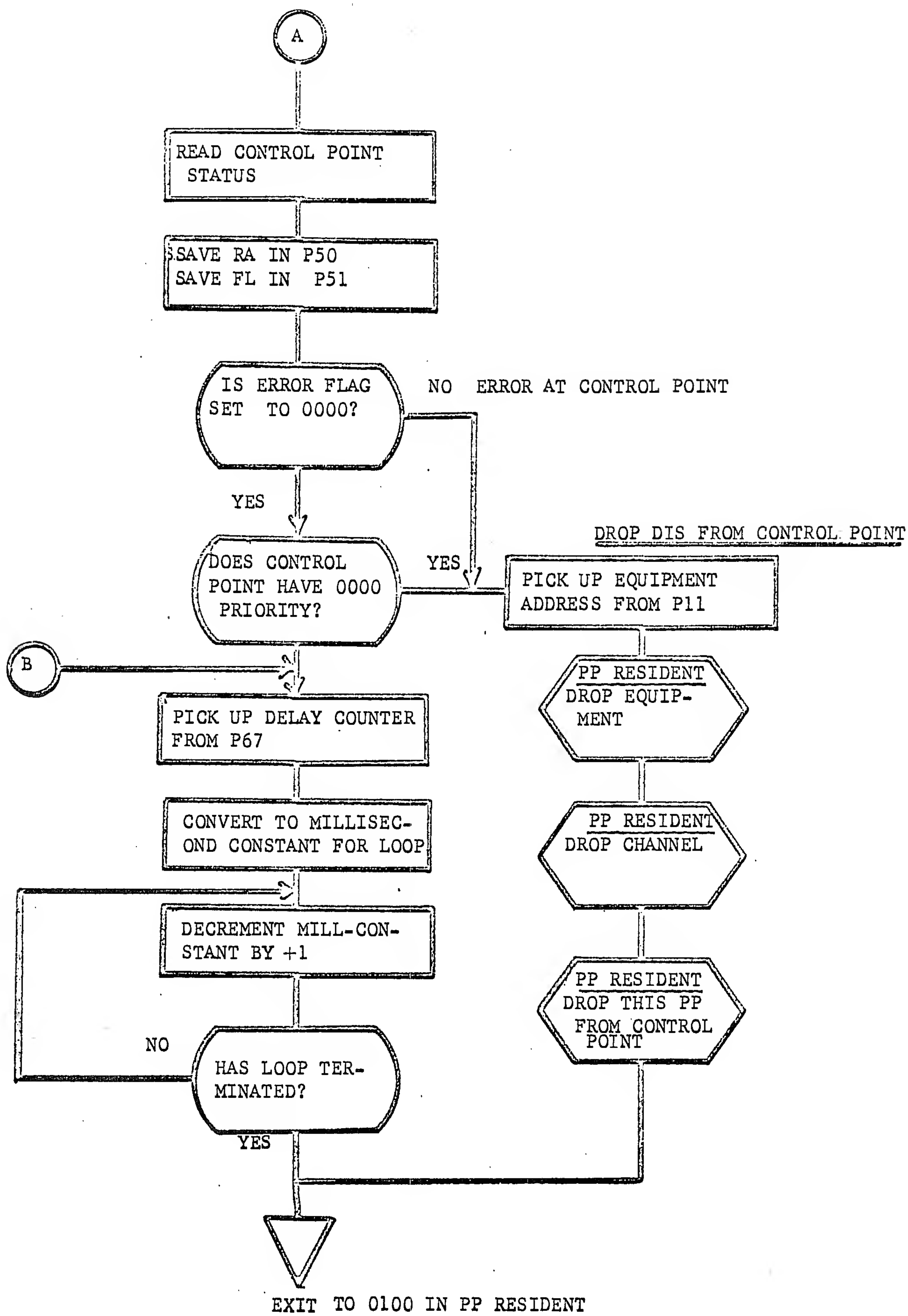
STATUS	CODE	CHAR
DISCONNECTED	04	D
EMPTY	05	E
FULL	06	F



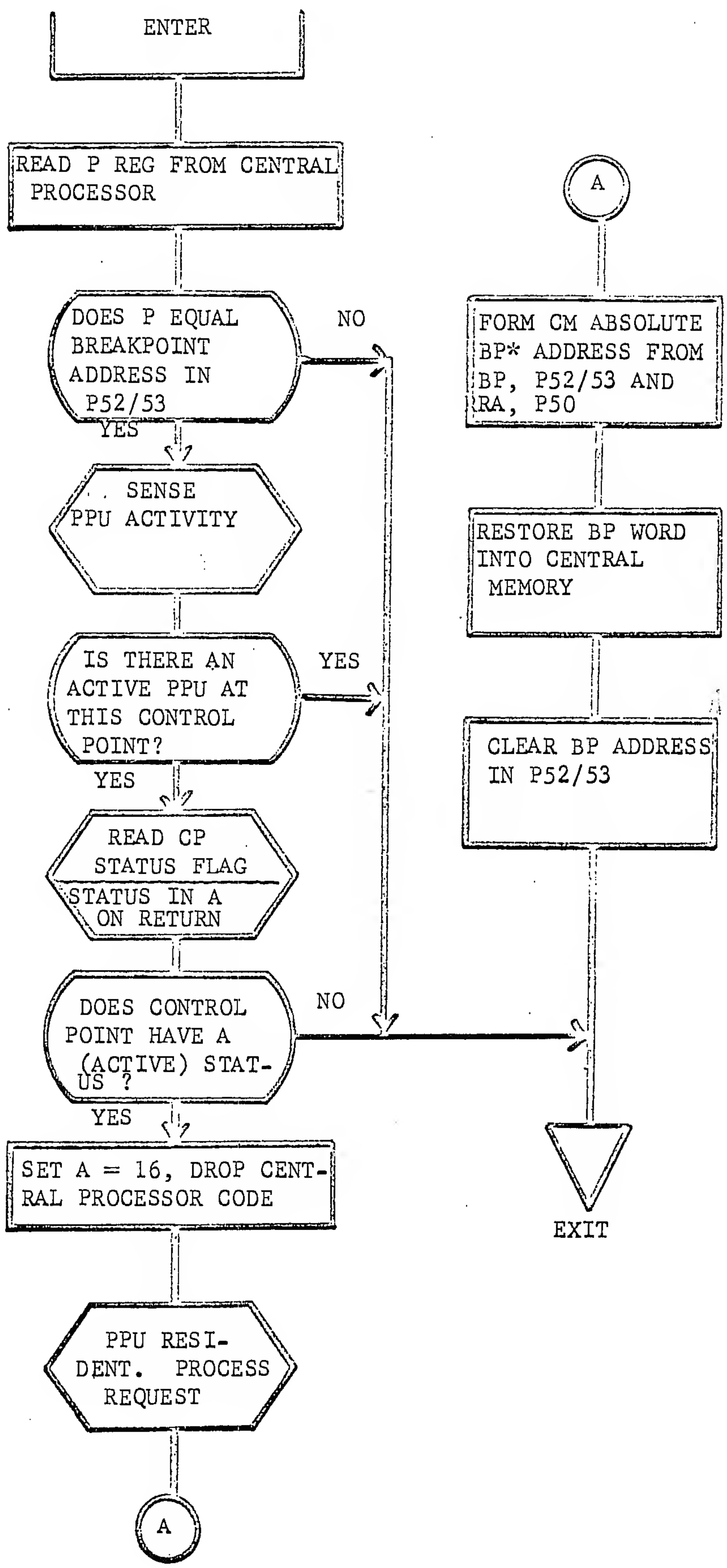

B
OUTPUT TWO STATUS CODES IN P02
HAVE FOUR CONSECTUTIVE CODES BEEN OUTPUT?
n NO
YES
OUTPUT TWO SPACES
HAVE ALL CHANNELS BEEN EXAMINED ?
NO
C
YES
EXIT

DIS - ADJUST DISPLAY PERIOD




A
READ CONTROL POINT STATUS
SAVE RA IN P50
SAVE FL IN P51
IS ERROR FLAG SET TO 0000?
NO ERROR AT CONTROL POINT
YES
DOES CONTROL POINT HAVE 0000 PRIORITY?
YES
DROP DIS FROM CONTROL POINT
PICK UP EQUIPMENT ADDRESS FROM P11
PP RESIDENT
DROP EQUIPMENT
PP RESIDENT
DROP CHANNEL
PP RESIDENT
DROP THIS PP FROM CONTROL POINT
B
PICK UP DELAY COUNTER FROM P67
CONVERT TO MILLISECOND CONSTANT FOR LOOP
DECREMENT MILL-CONSTANT BY +1
NO
HAS LOOP TERMINATED?
YES
EXIT TO 0100 IN PP RESIDENT

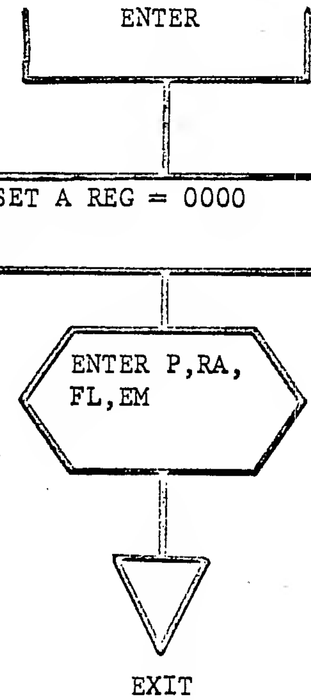
DIS - MONITOR BRAEKPOINT ADDRESS



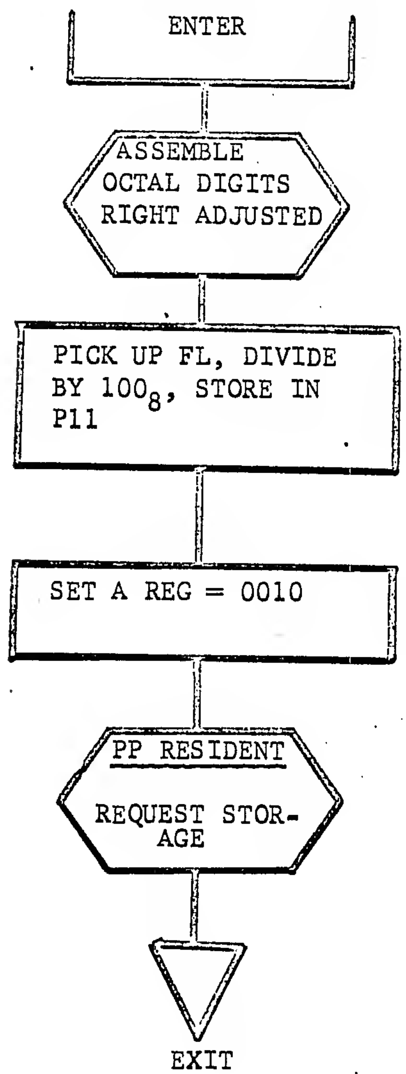
STATUS	CODE
WAITING	W
RECALL	X
ACTIVE	A
IN STACK	B to G

* BP Means BreakPoint

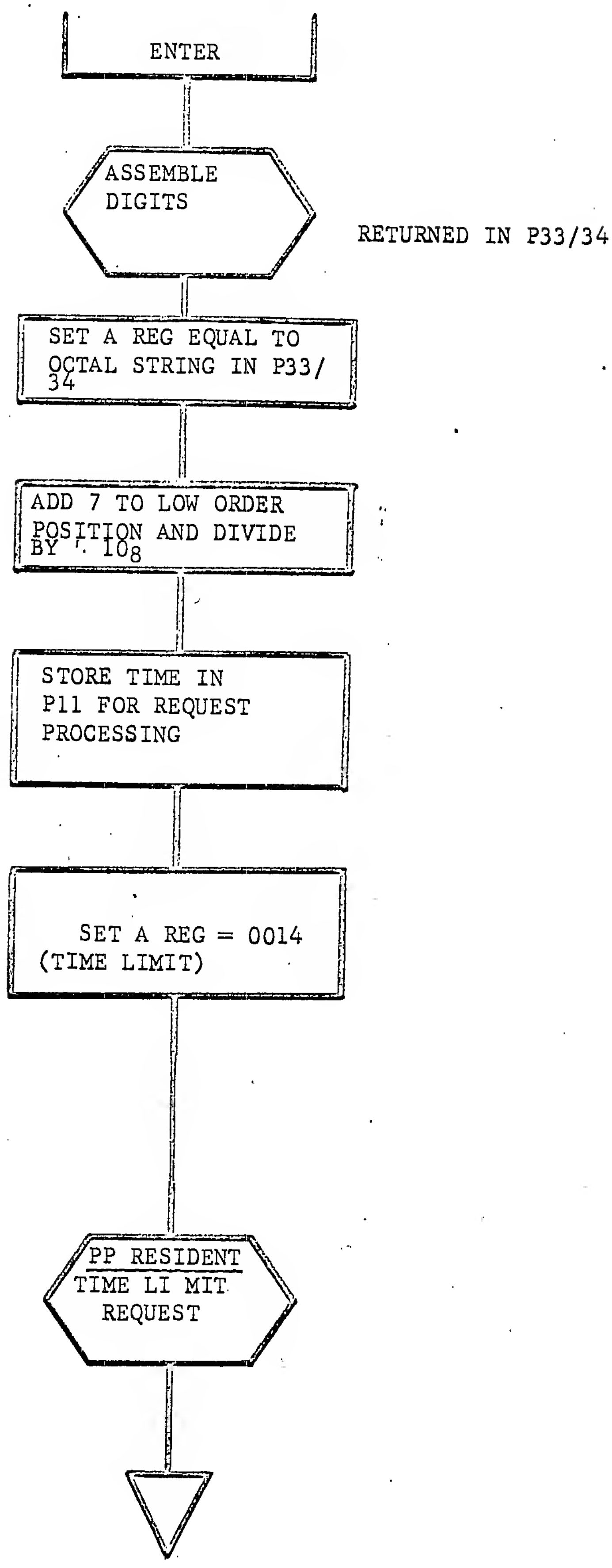
DIS - ENP REQUEST PROCESSOR



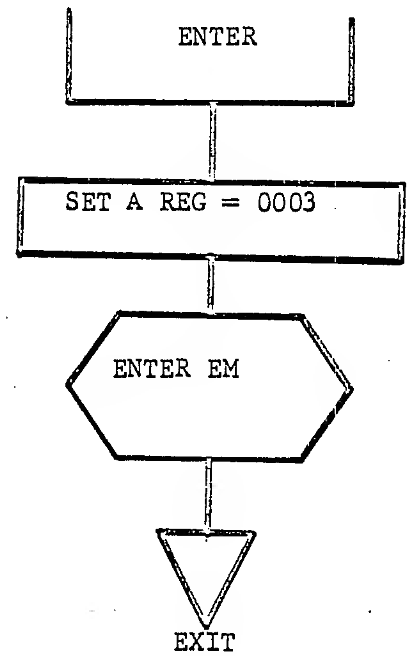
DIS - ENFL REQUEST PROCESSOR



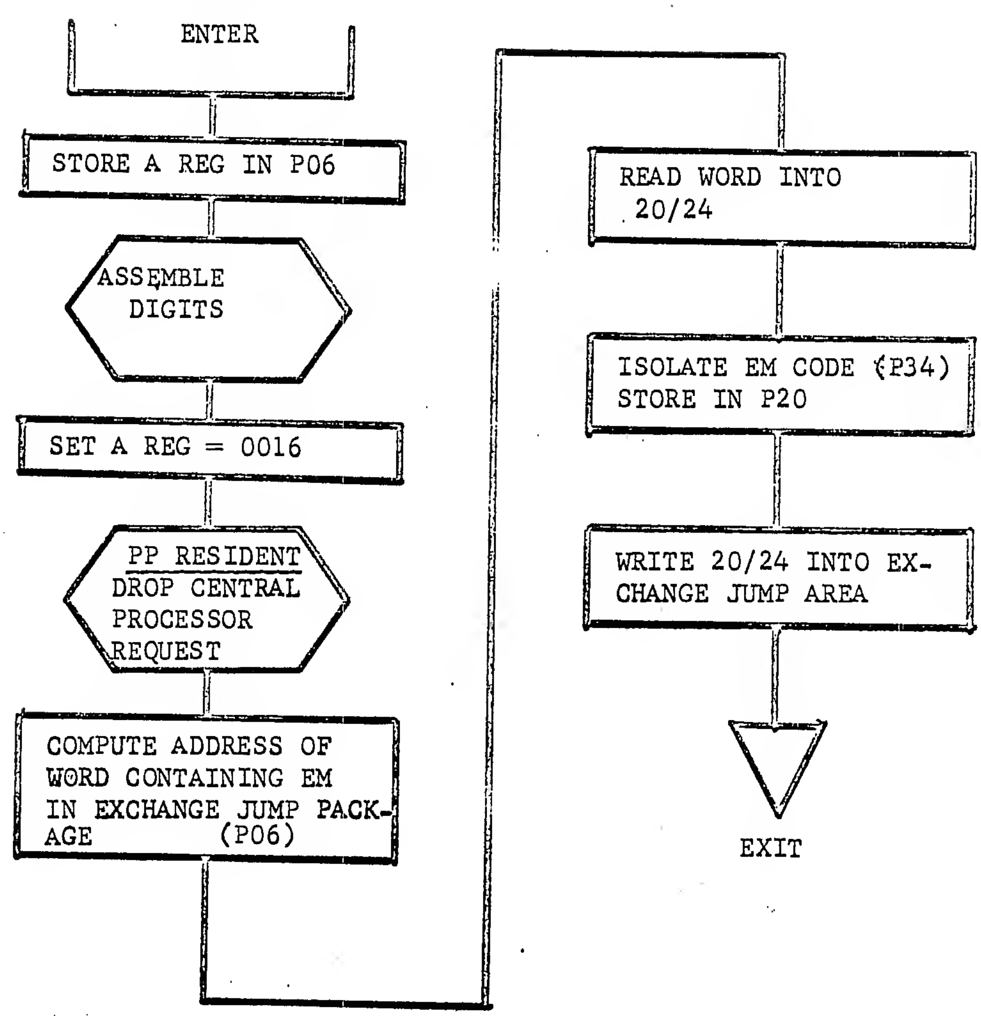
DIS - ENTL REQUEST PROCESSOR



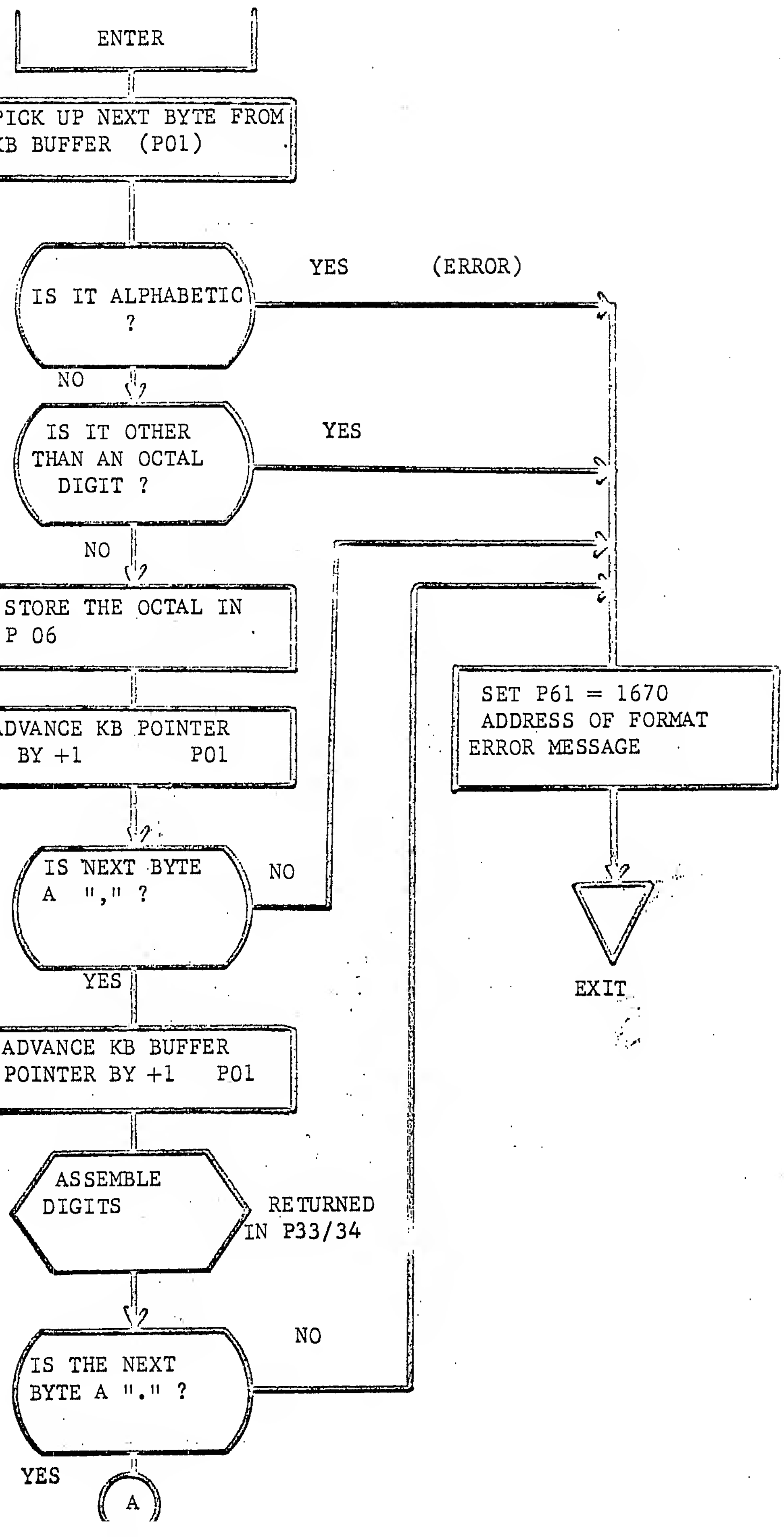
DIS - ENEM REQUEST PROCESSOR



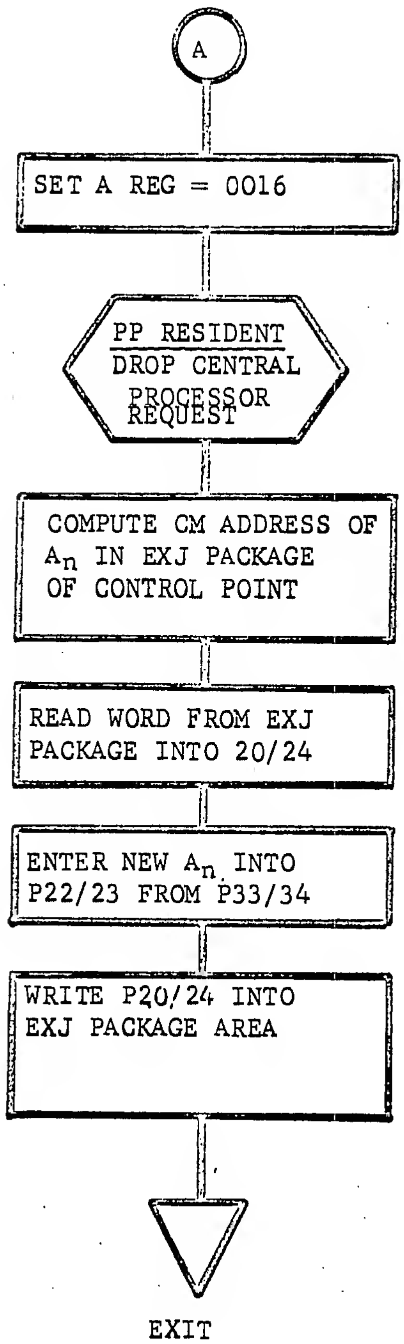
DIS - ENTER EM



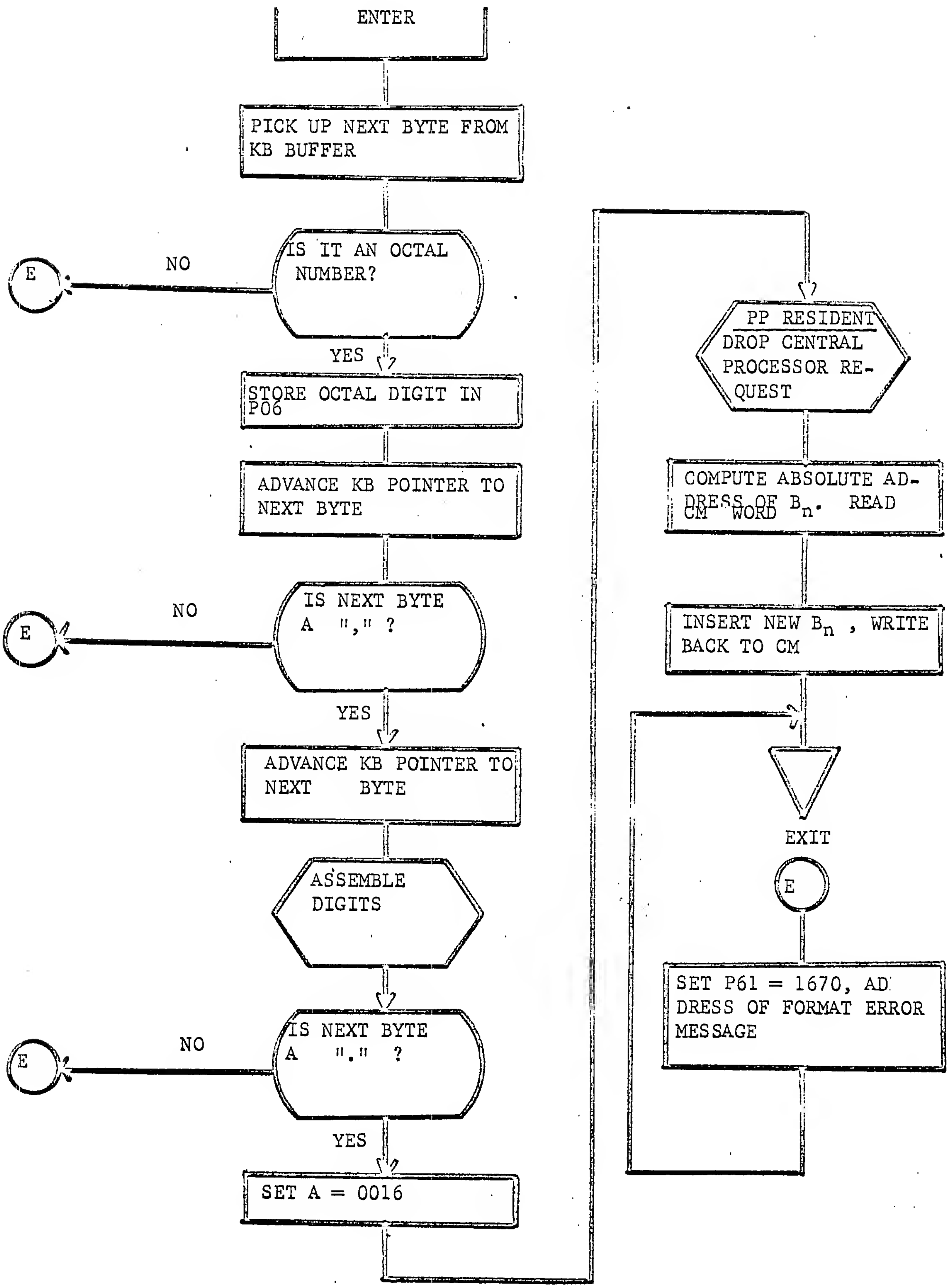
DIS - ENA REQUEST PROCESSOR



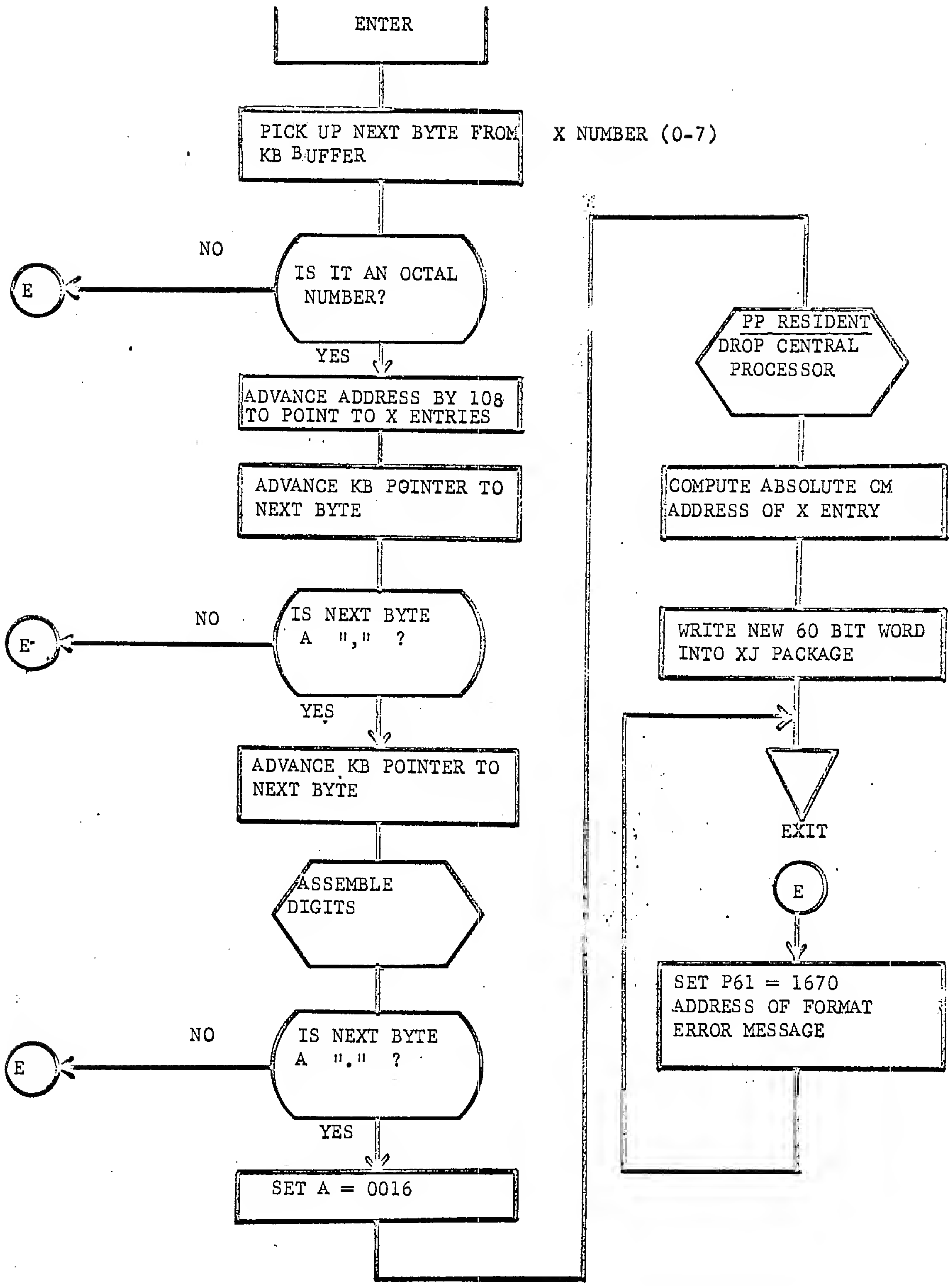
DIS - ENA REQUEST PROCESSOR (CONTINUED)



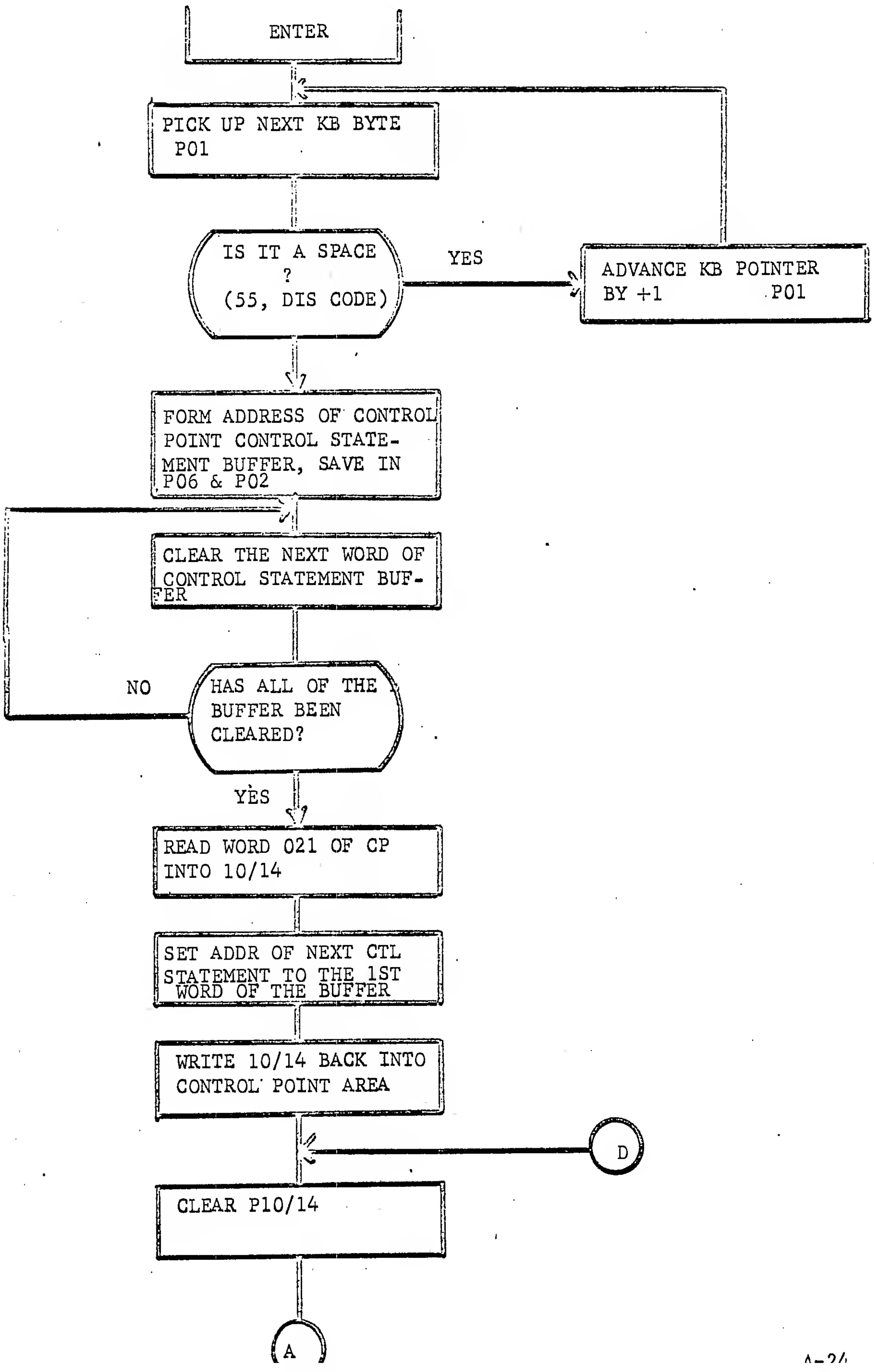
DIS - ENB REQUEST PROCESSOR



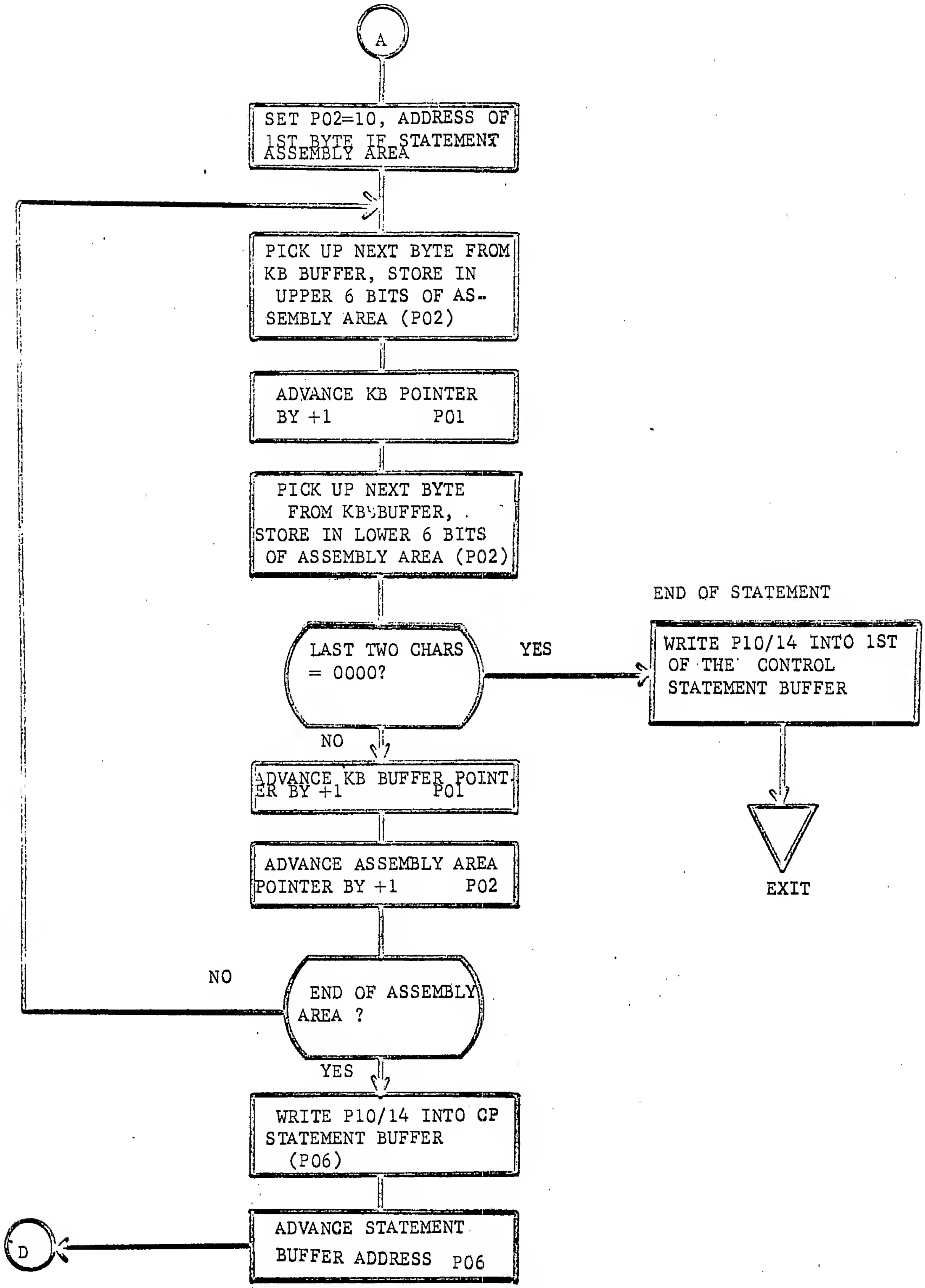
DIS - ENX REQUEST PROCESSOR



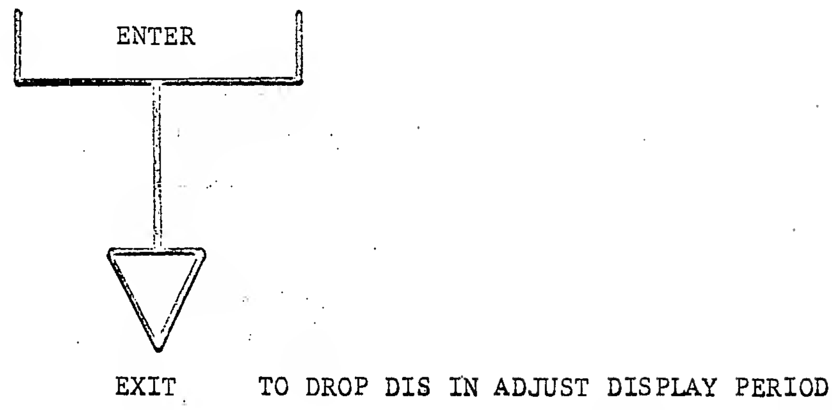
DIS - ENS REQUEST PROCESSOR



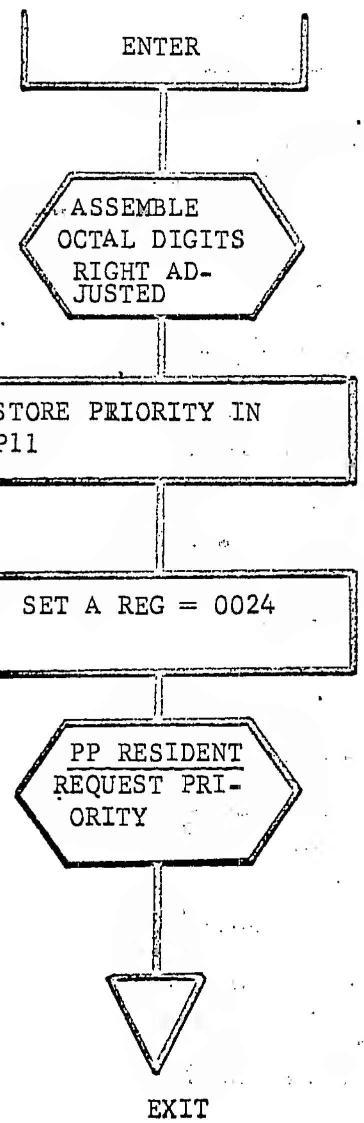
DIS - ENS REQUEST PROCESSOR (CONTINUED)



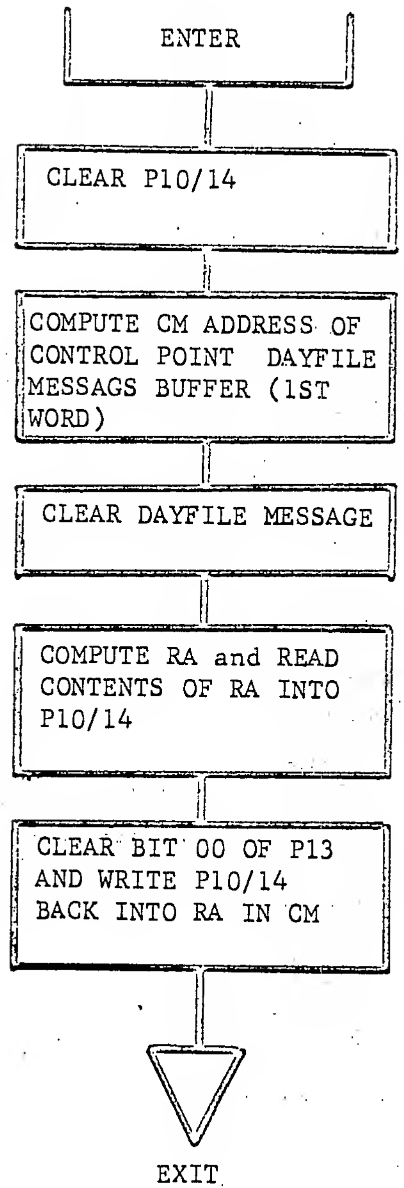
DIS - DROP REQUEST PROCESSOR



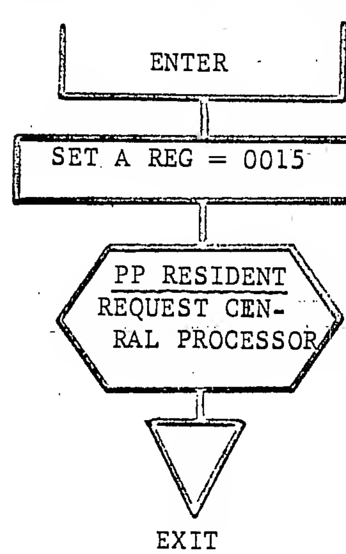
DIS - ENPR REQUEST PROCESSOR



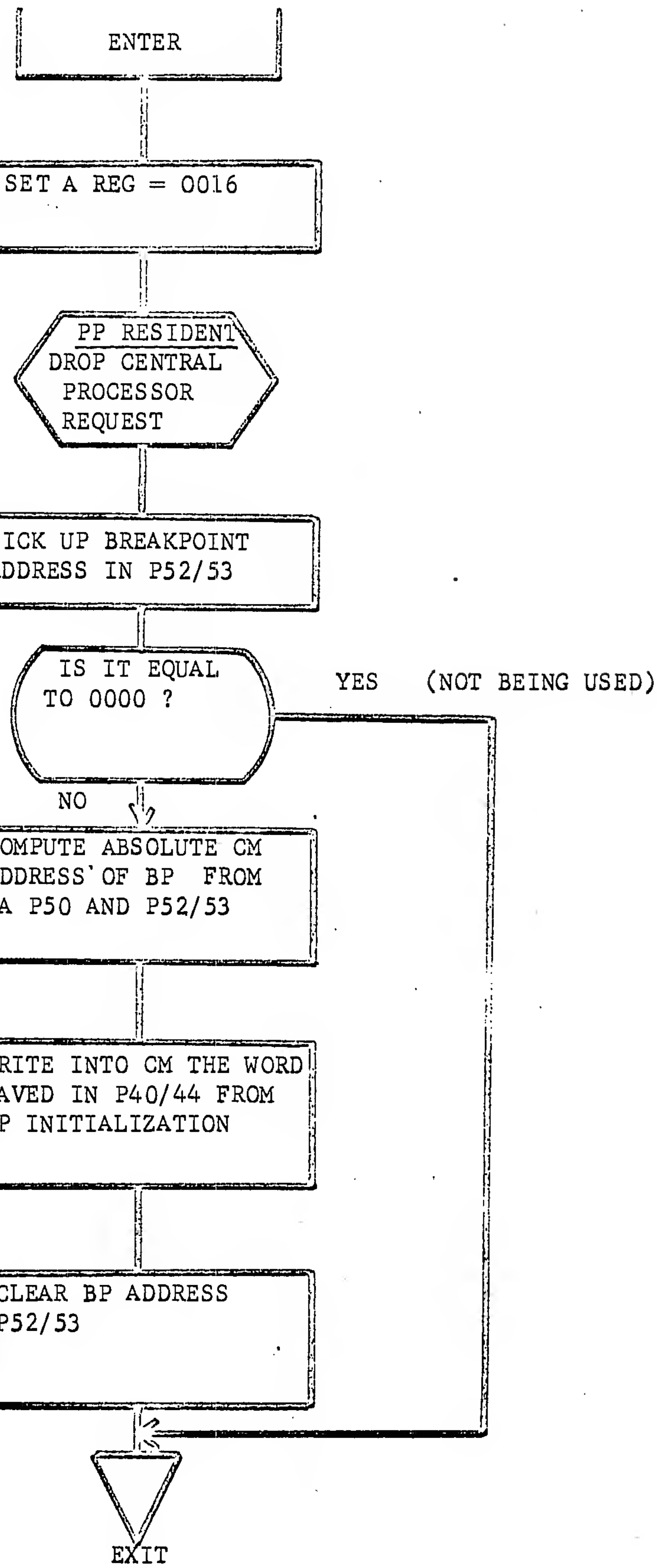
DIS - GO REQUEST PROCESSOR



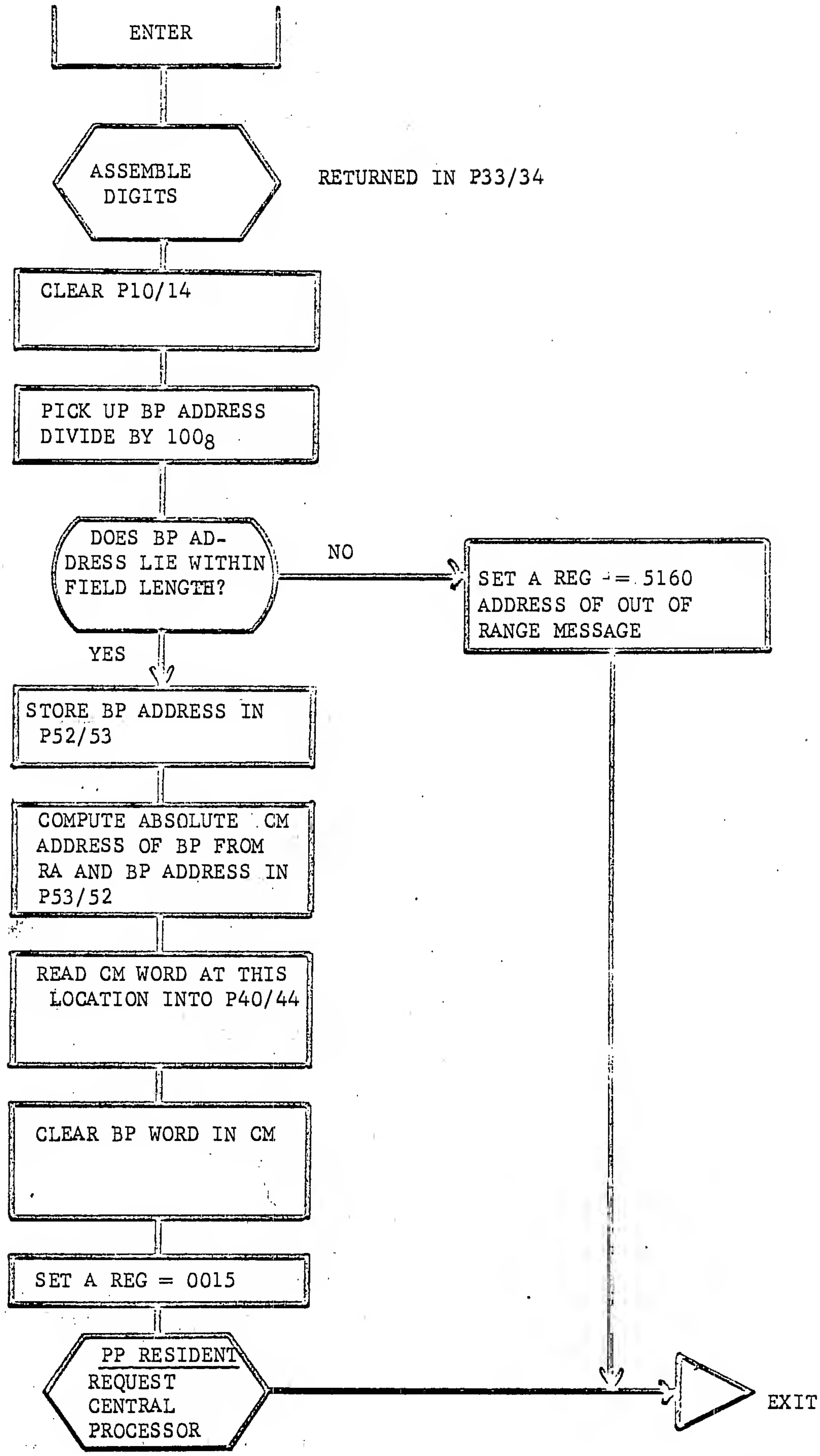
DIS - RCP REQUEST PROCESSOR



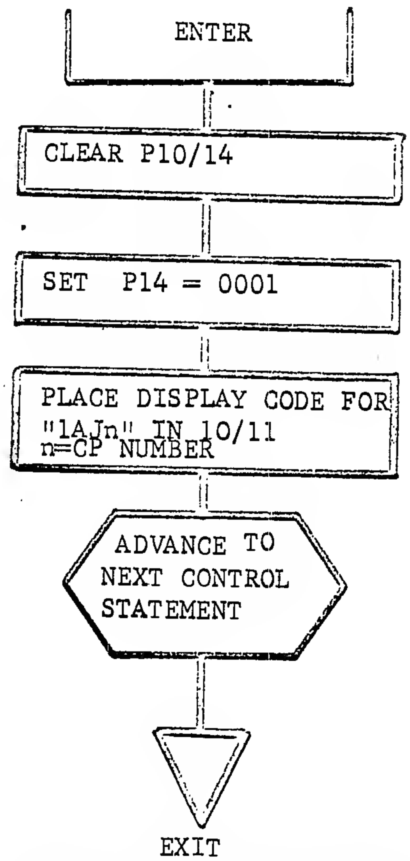
DIS - DCP REQUEST PROCESSOR



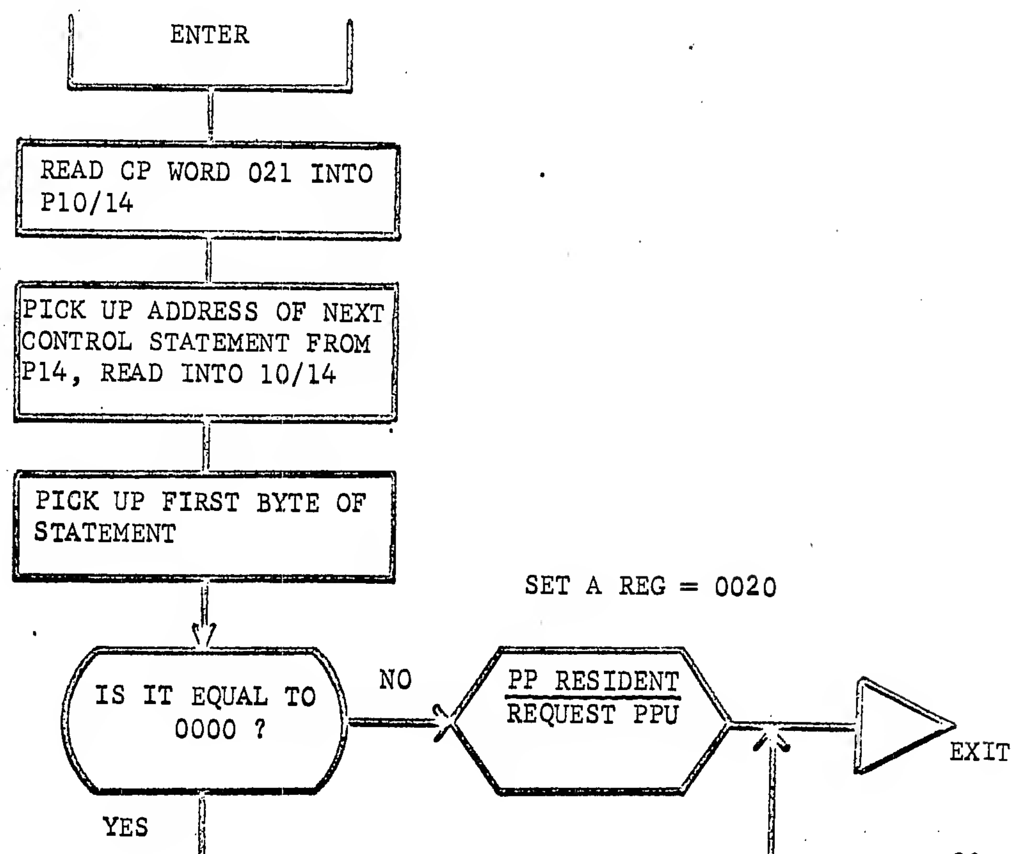
DIS - INITIATE BREAKPOINT REQUEST PROCESSOR



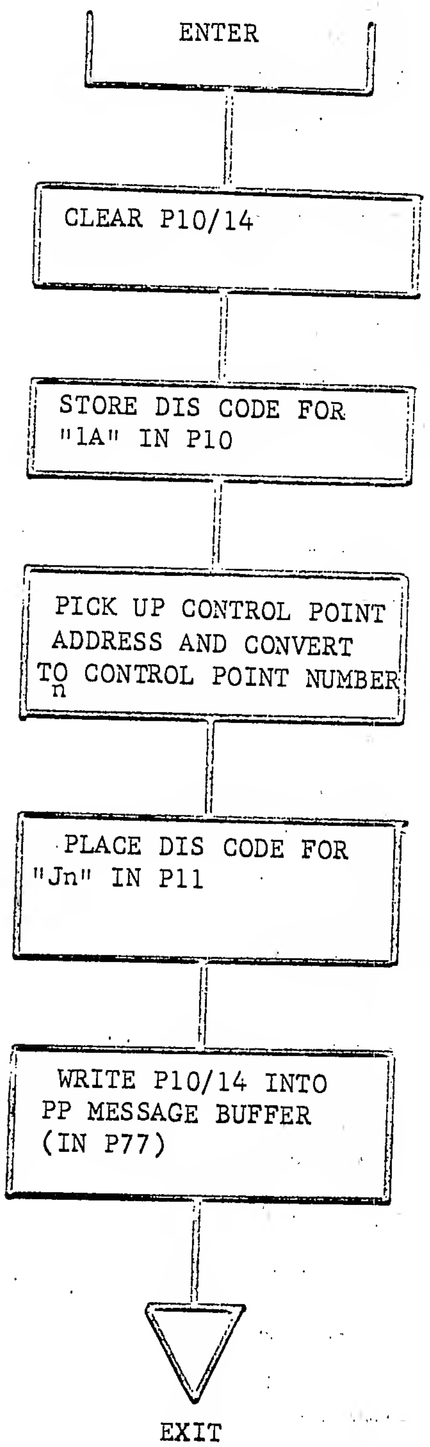
DIS - RSS REQUEST PROCESSOR



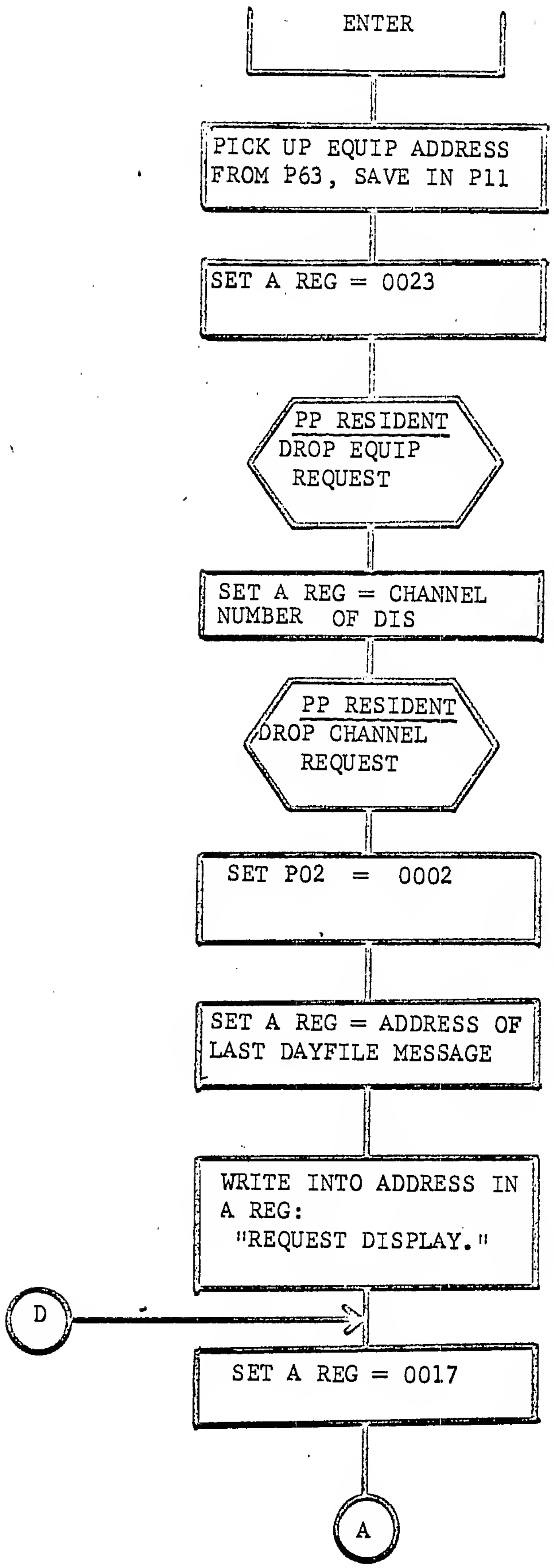
DIS - ADVANCE TO NEXT CONTROL STATEMENT

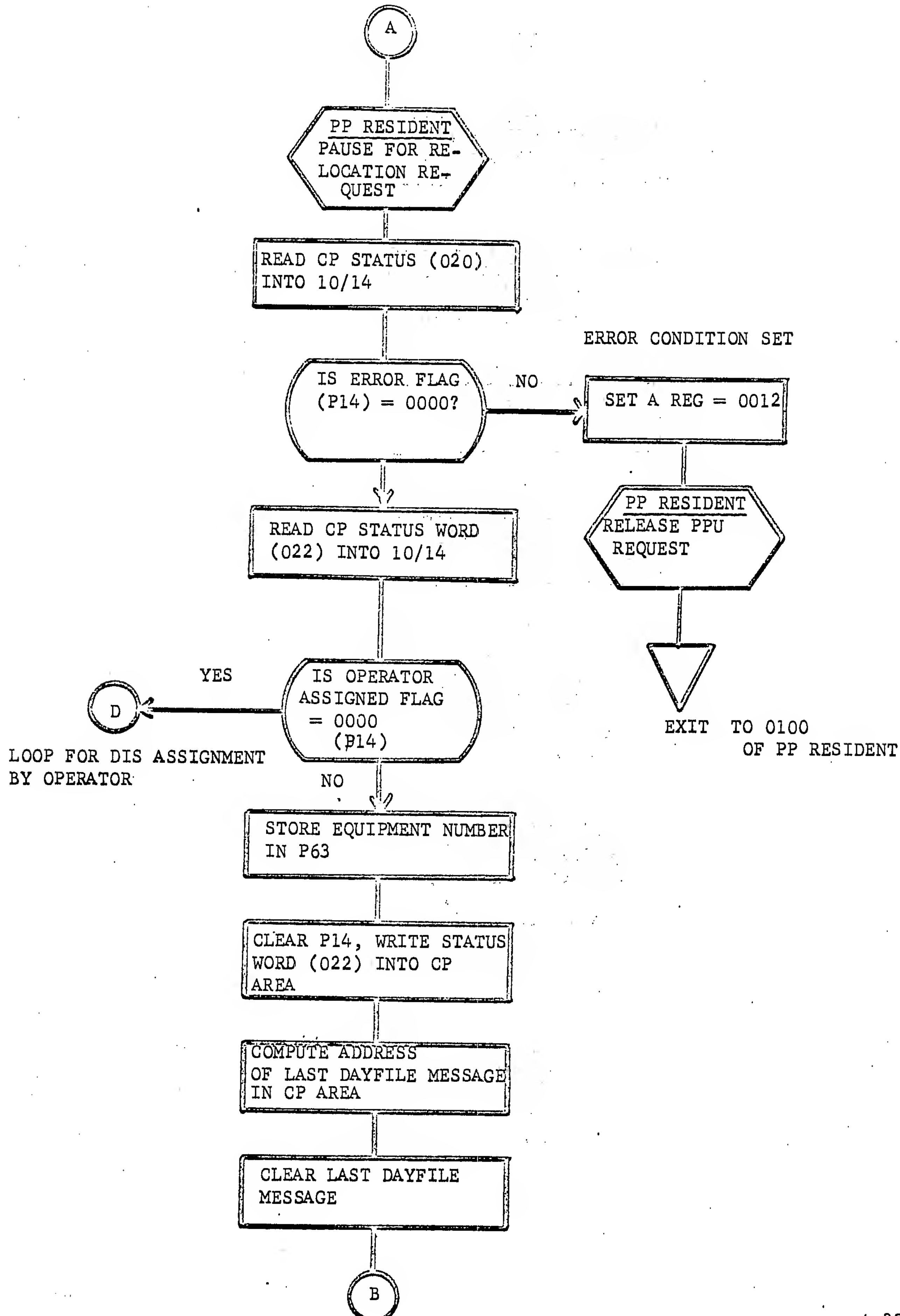


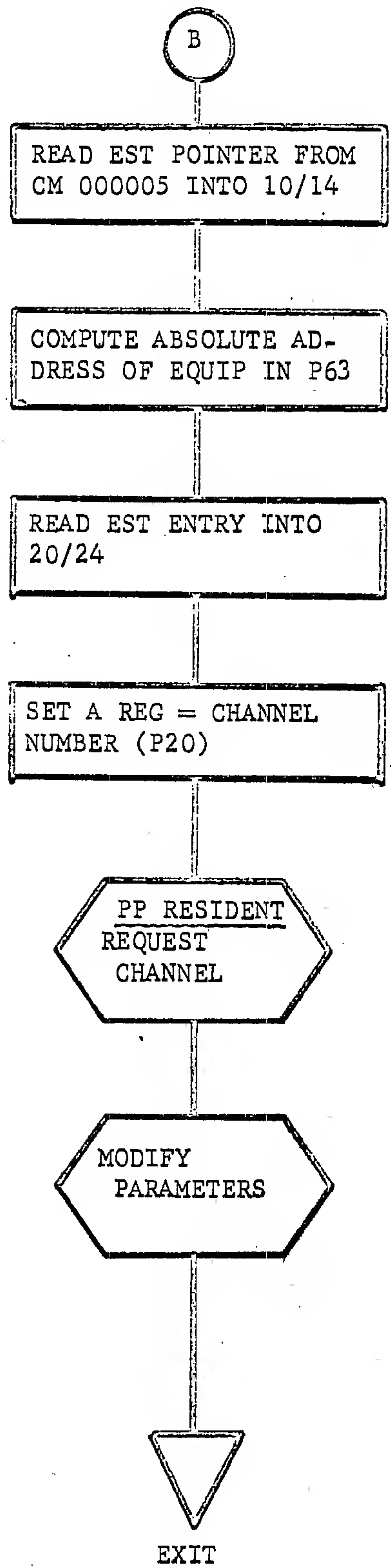
DIS - RNS REQUEST PROCESSOR



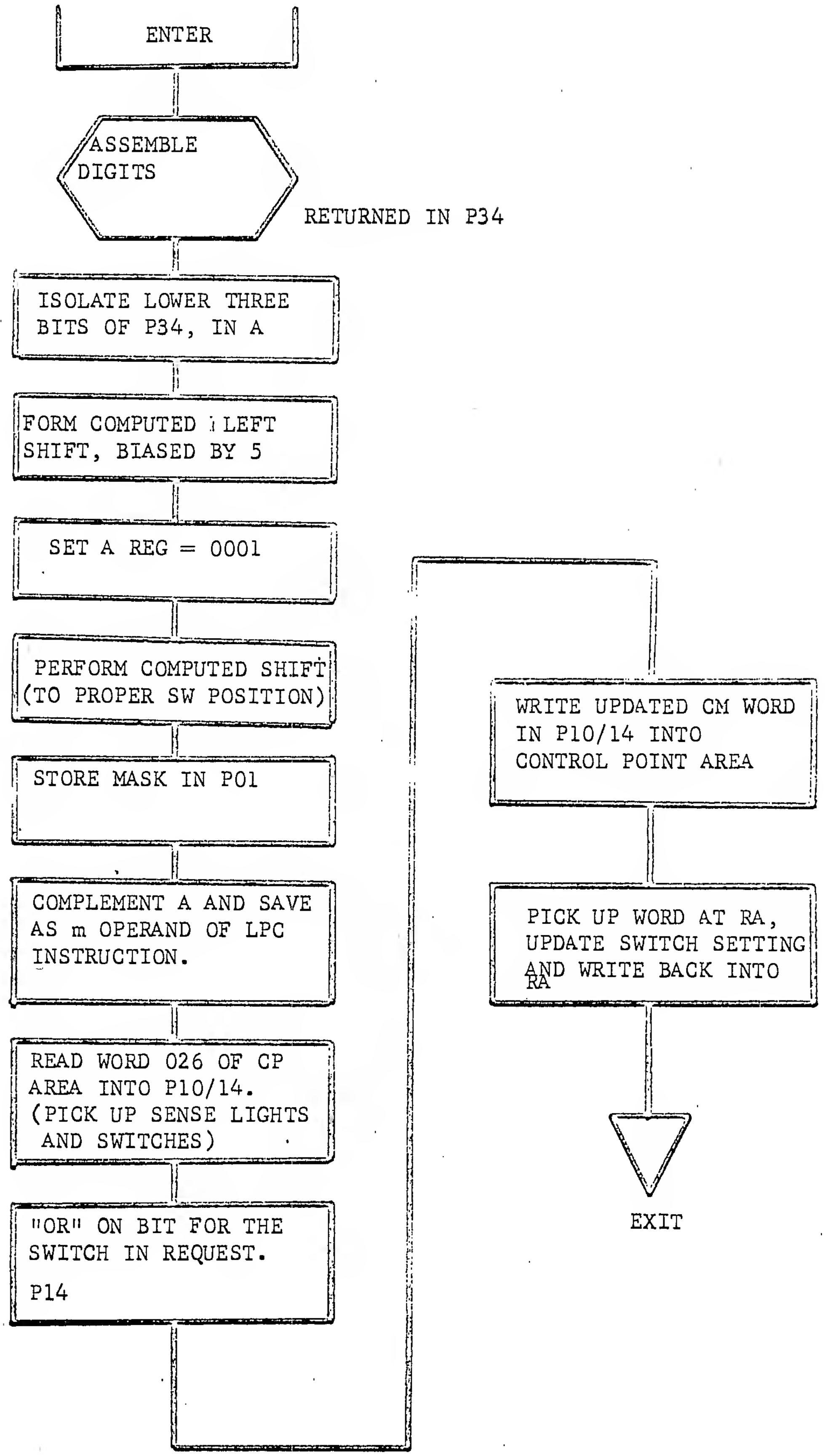
DIS - HOLD REQUEST PROCESSOR



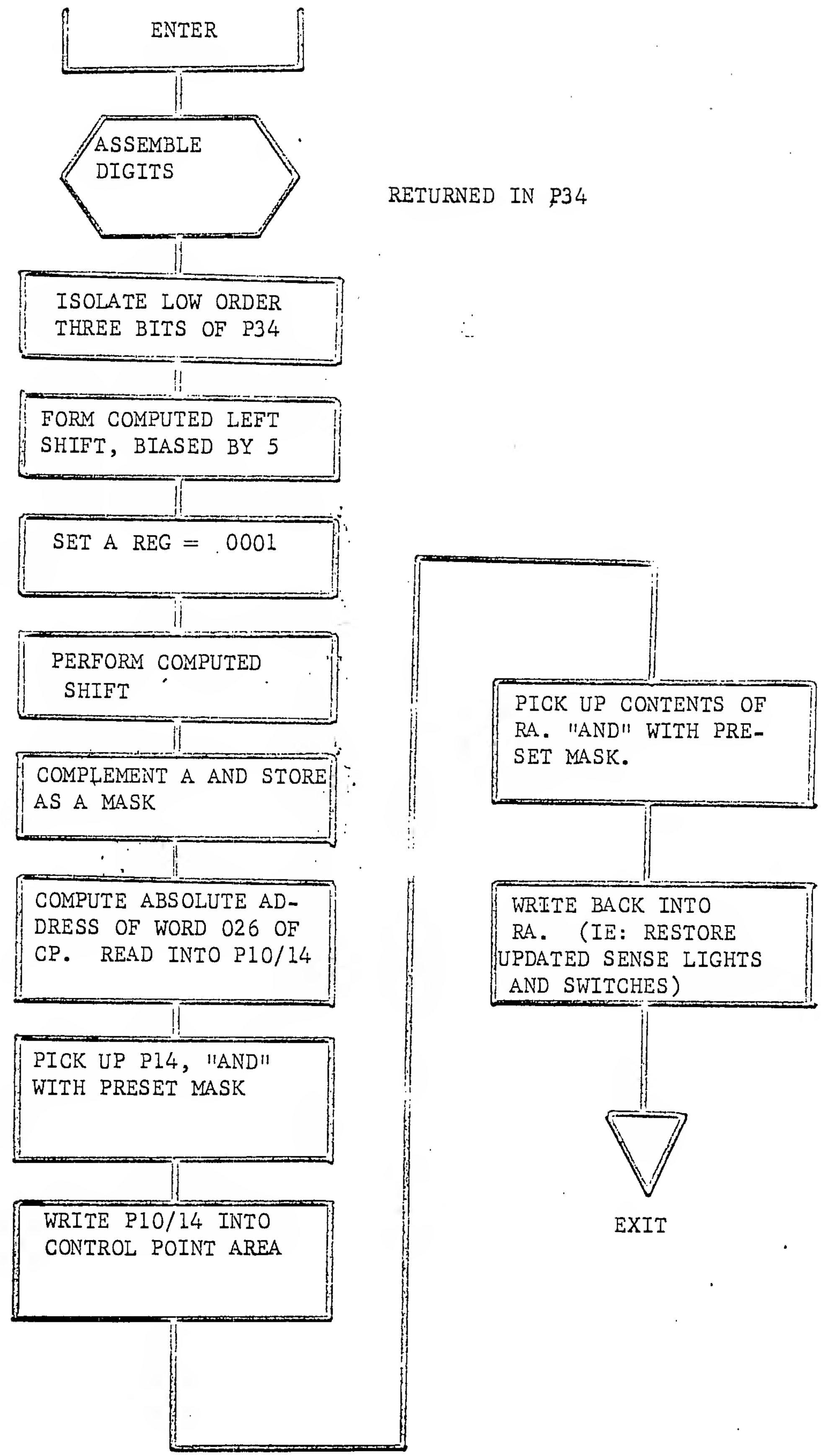

A
PP RESIDENT
PAUSE FOR RE-
LOCATION RE-
QUEST
READ CP STATUS (020)
INTO 10/14
ERROR CONDITION SET
IS ERROR FLAG
(P14) = 0000?
NO
SET A REG = 0012
PP RESIDENT
RELEASE PPU
REQUEST
EXIT TO 0100
OF PP RESIDENT
READ CP STATUS WORD
(022) INTO 10/14
YES
IS OPERATOR
ASSIGNED FLAG
= 0000
(P14)
D
LOOP FOR DIS ASSIGNMENT
BY OPERATOR
NO
STORE EQUIPMENT NUMBER
IN P63
CLEAR P14, WRITE STATUS
WORD (022) INTO CP
AREA
COMPUTE ADDRESS
OF LAST DAYFILE MESSAGE
IN CP AREA
CLEAR LAST DAYFILE
MESSAGE
B


B
READ EST POINTER FROM CM 000005 INTO 10/14
COMPUTE ABSOLUTE ADDRESS OF EQUIP IN P63
READ EST ENTRY INTO 20/24
SET A REG = CHANNEL NUMBER (P20)
PP RESIDENT
REQUEST CHANNEL
MODIFY PARAMETERS
EXIT

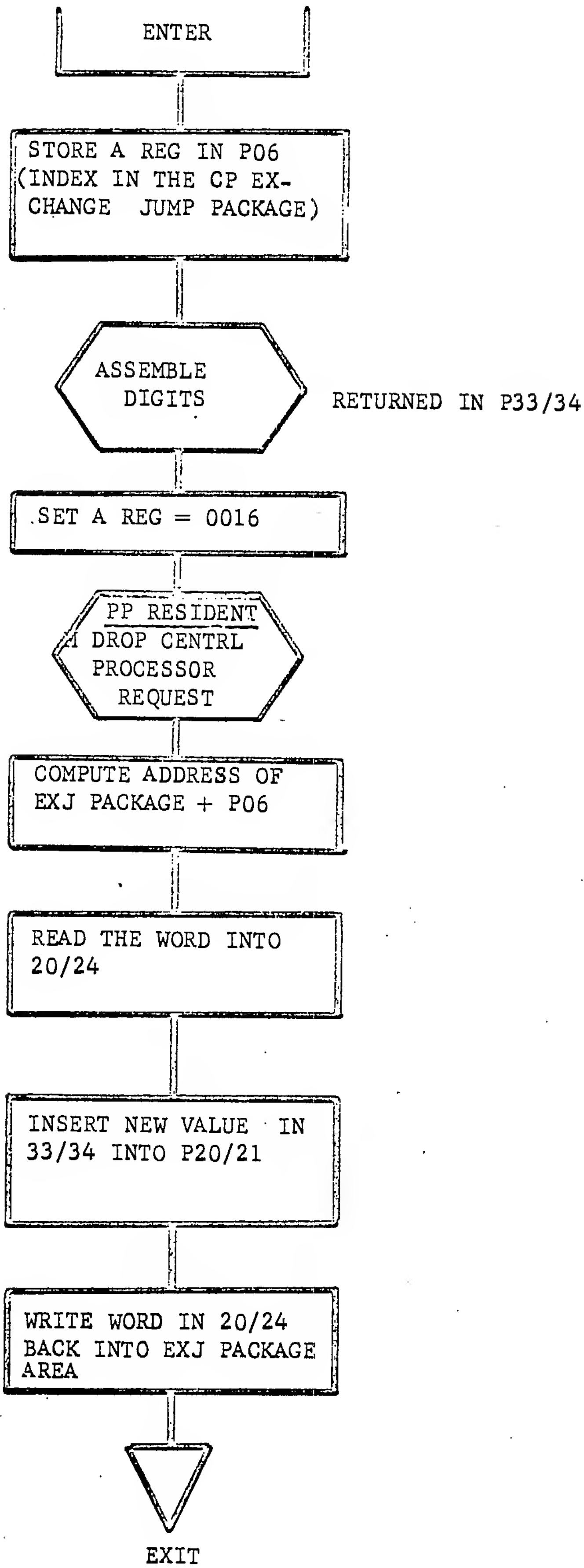
DIS - ONSW REQUEST PROCESSOR



DIS - OFFSW REQUEST PROCESSOR



DIS - ENTER P, FL, RA, EM



DIS - DISPLAY C, D, E, F, G

ENTER

SET P24 = ADDRESS OF 1ST ENTRY OF DISPLAY FIELD TABLE. (EACH TYPE OF DISPLAY HAS ITS OWN TABLE)

MODE	ADDRESS
C	1470
D	1520
E	1550
F	1600
G	1630

NOTE: EACH TABLE HAS FOUR ENTRIES, TWO PPU WORDS EACH, CORRESPONDING TO FIELD.

SET A REG = ADDRESS OF DISPLAY FORMAT PROCESSOR:

MODE	ADDRESS
C	3000
D	3000
E	3000
F	2600
G	2600

NOTE:
(3000)=4GRPS OF 5DIGTS
(2600)=5GPS OF 4DIGTS

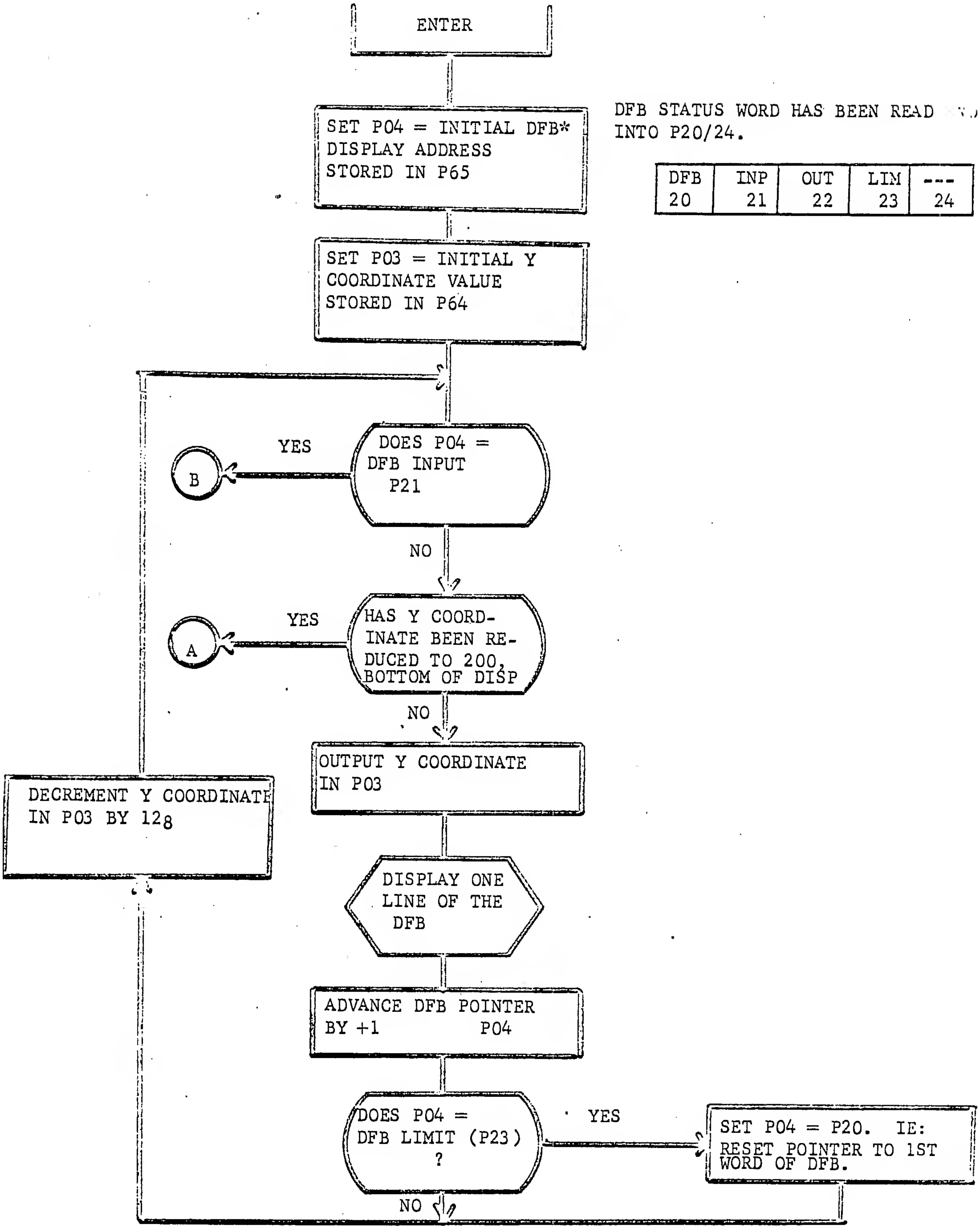
DISPLAY STORAGE

EXIT

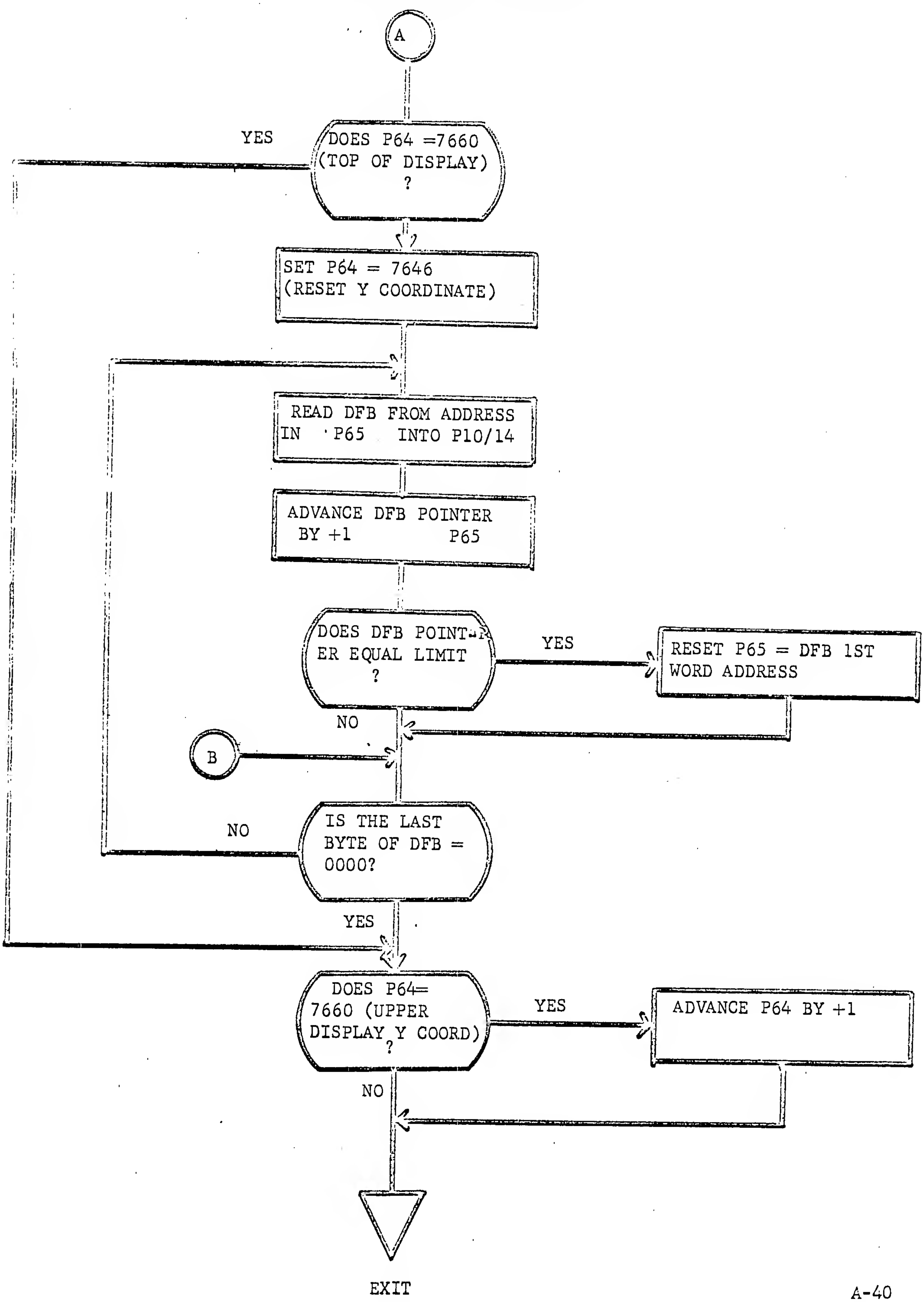
DIS - DISPLAY DAYFILE

DFB STATUS WORD HAS BEEN READ INTO P20/24.

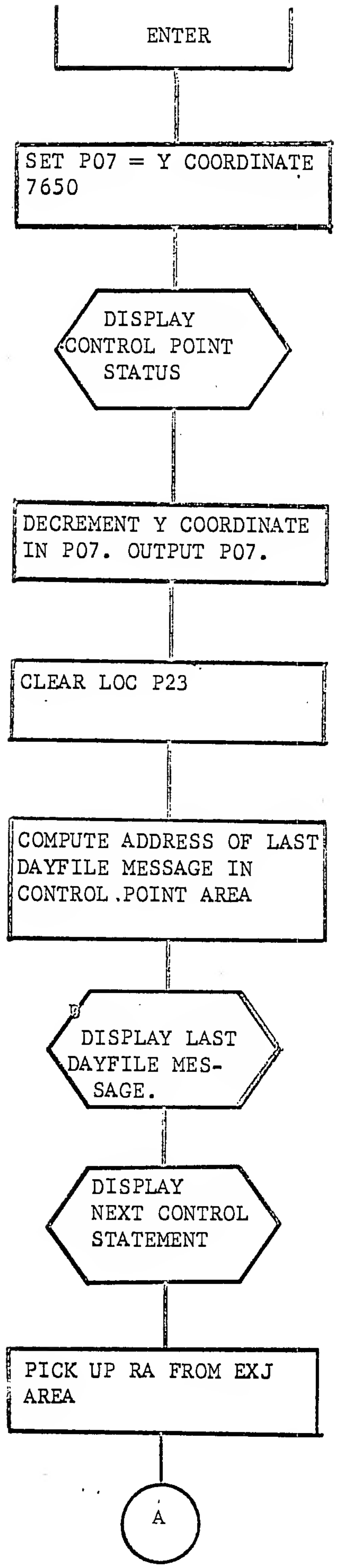
DFB	INP	OUT	LIM	---
20	21	22	23	24



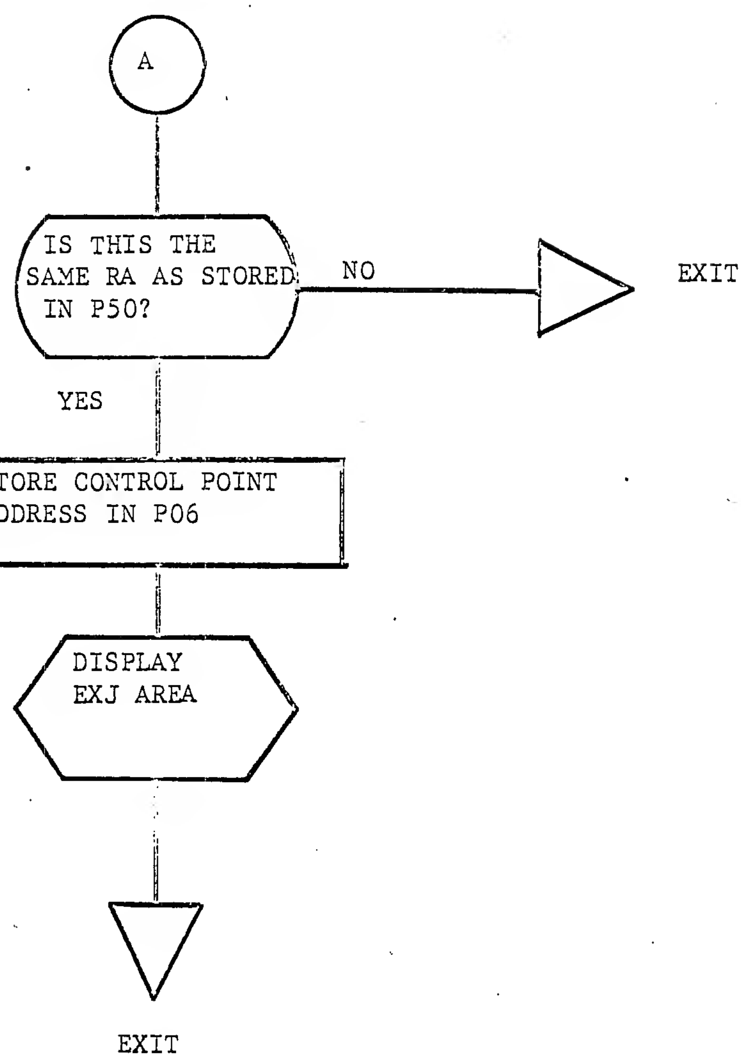
DIS - DISPLAY DAYFILE (CONTINUED)



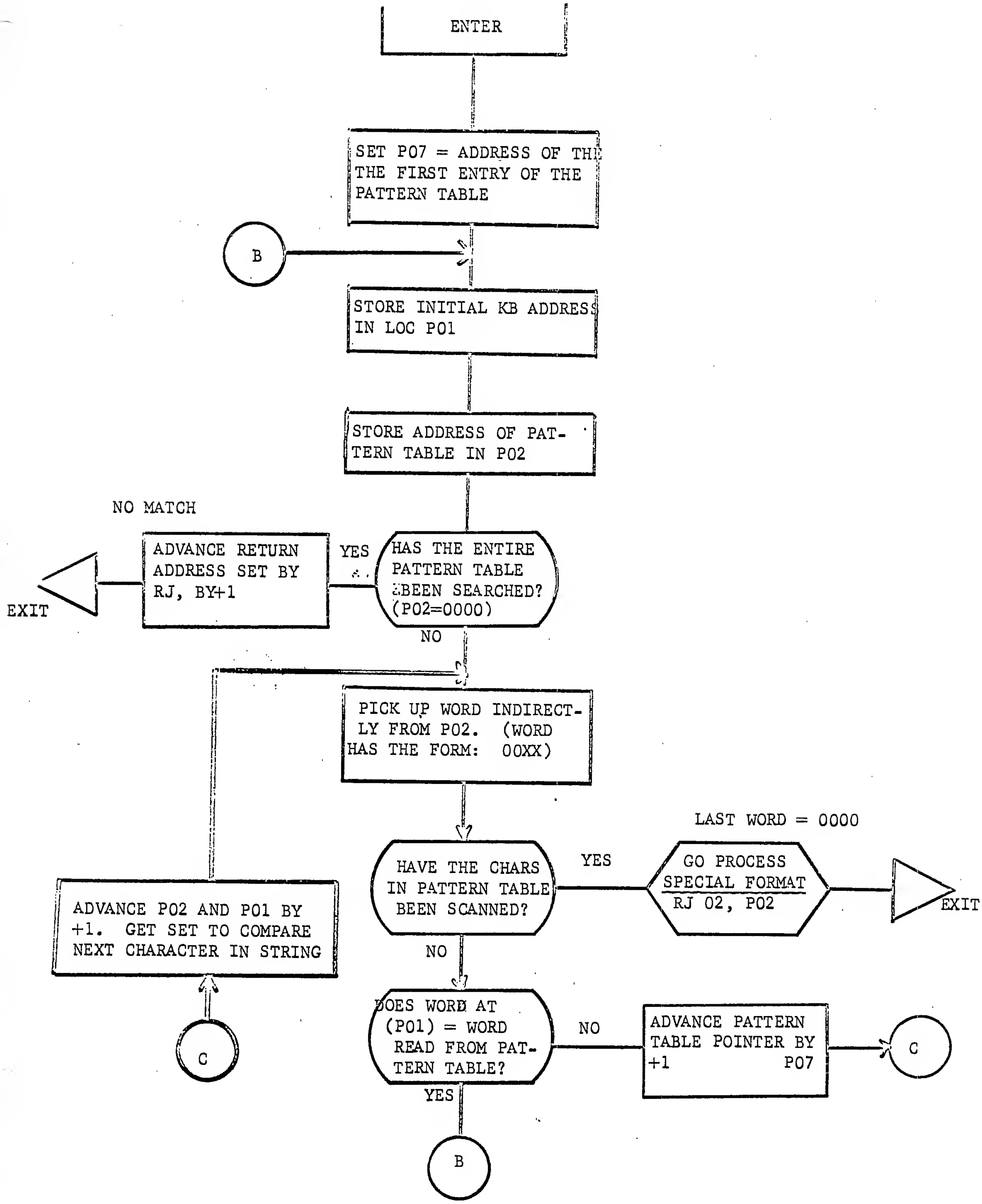
DIS - DISPLAY B



DIS - DISPLAY B (CONTINUED)



DIS - SEARCH FOR SPECIAL FORMAT



CONTROL DATA CORPORATION

Development Division - Applications

DISK ROUTINES AND OVERLAYS

Chippewa Operating System

11/1/65

Disk Routines and Overlays

Contents

DISK ROUTINES AND OVERLAYS

Introduction

In the Chippewa Operating System, there is no single system element used to perform disk operations for all other elements of the system. Instead, each system element performs its own disk operations. This, while requiring additional coding for each of the system elements using the disk, eliminates the need for a request queueing and priority scheme required by the use of a single system element to process all disk operations. In addition, the housekeeping required by a disk subroutine in one system element can overlap, to some extent, a disk operation being performed by another system element. Among the system elements which perform disk operations are:

- peripheral processor resident (reads transient programs from the disk library)
- MTR (writes the contents of the dayfile buffer to the disk)
- some transient programs (read overlays from the disk)

Disk operations for external users are performed via the overlays 2WD (write disk), 2RD (read disk), and 2BD (backspace disk). These overlays are called by CIO when a disk operation is requested by a central processor program. In addition, these overlays are used by certain transient programs to perform disk operations. Thus, 1LJ and 1LT call 2WD when loading jobs from the card reader and a tape unit, respectively, while 1DJ and 1TD call 2RD when transferring job output to the printer or a tape unit.

Regardless of where in the system they are performed, disk operations are similar: this discussion will therefore be limited to the overlays 2WD, 2RD, and 2BD. Before discussing these routines a short review of the physical characteristics of the 6603 disk file is in order.

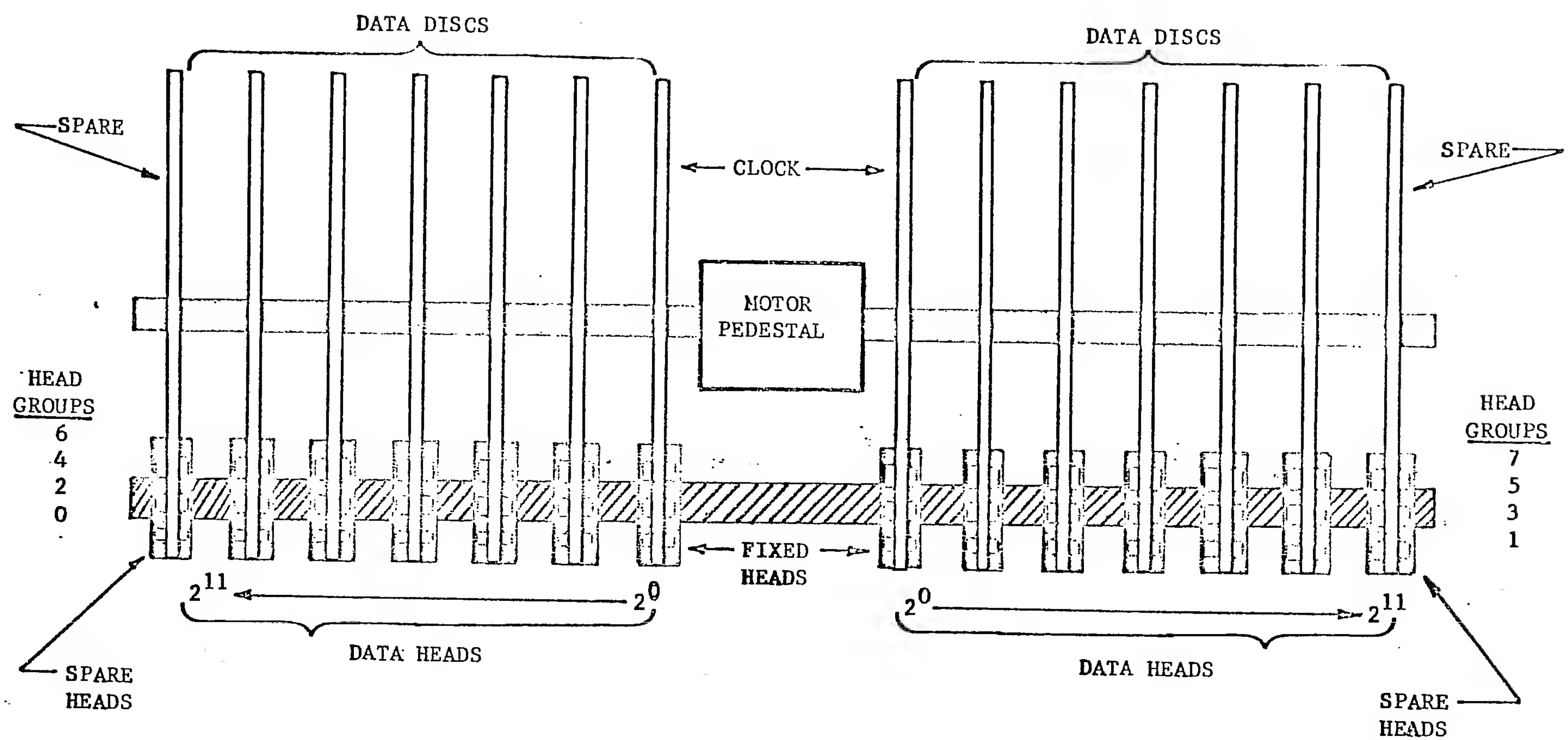
6603 Disk File: Description and Organization

The 6603 Disk File contains fourteen disks, each coated on both sides with magnetic oxide. Thus, there are a total of twenty-eight recording surfaces. On two of these surfaces timing tracks are recorded, two are used for spares, and twenty-four are used for recording data (see figure 1). All fourteen disks are mounted (in a vertical plane) on a common axis and rotate at a speed of approximately 900 revolutions per minute. Twelve of the data surfaces are on the right side of the unit, and twelve are on the left. Information is recorded on the disk in 12-bit bytes: each bit in a 12-bit byte is recorded on a separate disk surface.

Associated with each disk surface is a set of four read/write heads (see figure 2). An assembly consisting of a rocker arm and a head bar fits between each pair of facing disk surfaces. The head bar holds two sets of four heads, one set for each of the two facing surfaces. The read/write heads are mounted on this head bar in a fixed position relative to each other. The rocker arm-head bar assemblies for all disks mount on a common bracket which can be rotated. This rotation moves all the head bars simultaneously (with the exception of the heads accessing the timing track surfaces: these heads are fixed).

The disk surface is divided into four zones. A <u>zone</u> is that portion of the disk surface transversed by one of the four heads associated with that surface as the head (on its head bar-rocker arm assembly) moves through its maximum angular rotation. A byte may be written on the twelve data surfaces on the right side of the disk file or on the twelve data surfaces on the left side of the disk file: on either side, a byte may be written in any one of four zones. On each side of the disk file and for each zone on side, a single set of twelve read/write heads are used to record a byte (see figure 1). This set of twelve heads is called a <u>head group</u>. There are four head groups for each of the two sets of twelve disk surfaces: a total of eight head groups.

Each zone contains 128 <u>tracks</u>. A track is the recording path available to a given head group in a given position as the disk makes a complete revolution. To move from one track to another requires a physical



ALL HEADS (EXCEPT FIXED HEADS) MOVE TOGETHER

6603 DISC FILE

Figure 1

movement, or repositioning, of the head bar-rocker arm assemblies. At a given position, each head group accesses the same track in its zone. Thus, if head group 2 is positioned to track 125, the other 7 head groups are also positioned to track 125.

Tracks are divided into sectors: a sector is the smallest addressable segment of a track. There are 128 sectors in each of the tracks in the two outer zones. In the two innermost zones, there are only 100 sectors per track because of the reduced track length near the center of the disk compared to the track length available near the outside edge. A sector contains 351 bytes (each bit in a byte is recorded in one of 12 corresponding sectors across 12 disk surfaces). The first four bytes recorded are reserved for use by the controller: They provide a time lag between consecutive sectors and contain all zero bits. After the last data byte has been written, the controller writes a longitudinal parity byte. The sector format is illustrated in figure 3. Of the 351 bytes in a sector, then, five are used by the controller: The remaining 346 bytes may be used for data. Normally, 320 bytes (the equivalent of 64 central memory words) are used for data.

The number of words read from or written to the disk is solely a function of the word count specified in the IAM or OAM instruction. It is possible to read or write more than one sector at a time; it is possible to read or write in the group switch gap; it is possible for a read or write to wrap around on the same track. A read or write operation always begins at the beginning of a sector. When a write is initiated, the disk controller inserts four zero bytes before the data and inserts a parity byte after the last data byte. (The parity byte is not necessarily in the last byte position in a sector.) When a read is initiated, the controller assumes that the first four bytes are zero bytes, and does not pass these on to the data channel. When the word count in a read has been reduced to zero, the controller assumes that the next byte to be read is the parity byte. Thus, any attempt to read a number of bytes different than the number of bytes written will invariably create problems due to the interpretation of zero bytes and parity bytes as data and vice versa. For this reason, regardless of the amount of data to be recorded, a fixed number of bytes is written in each sector,

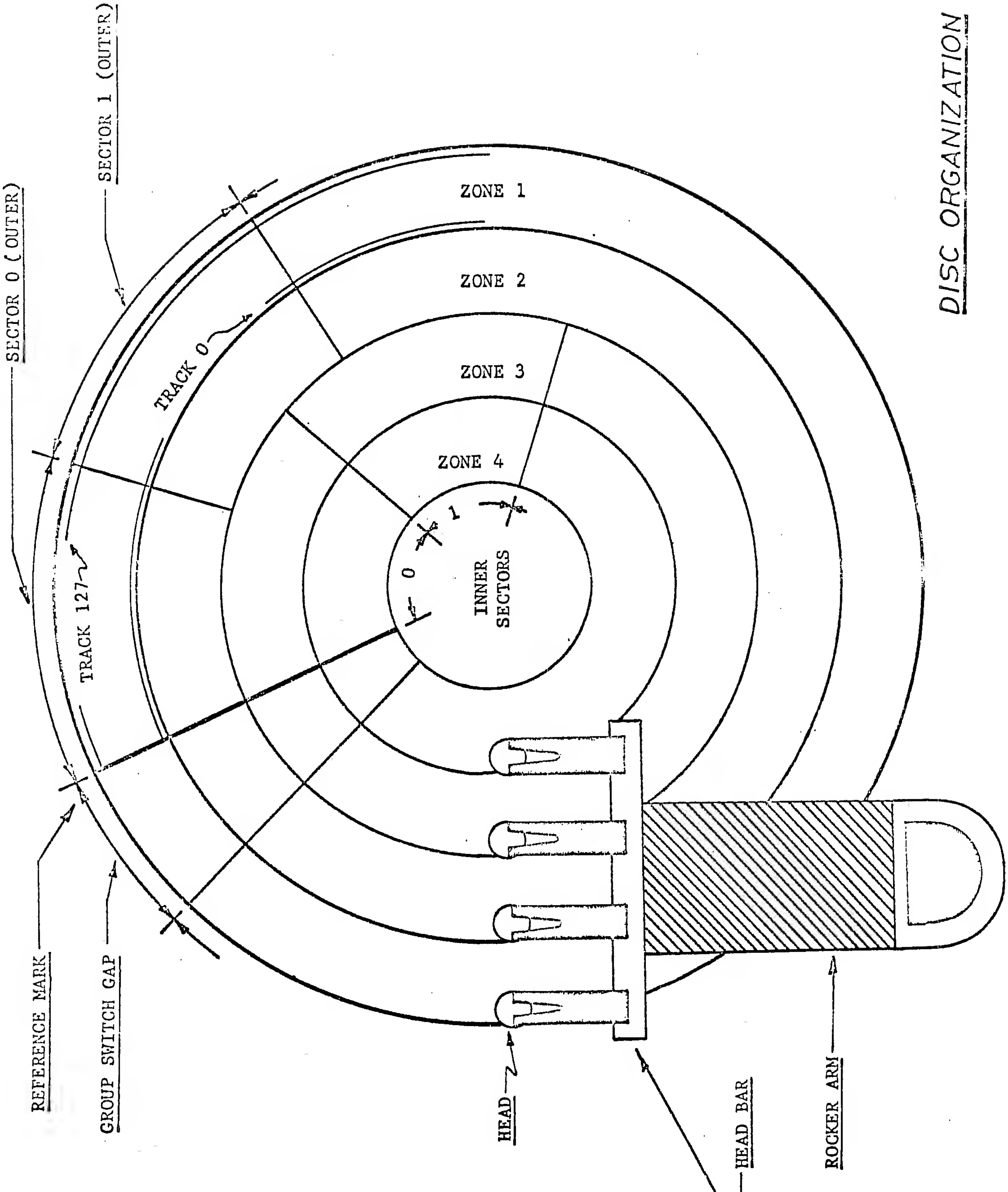
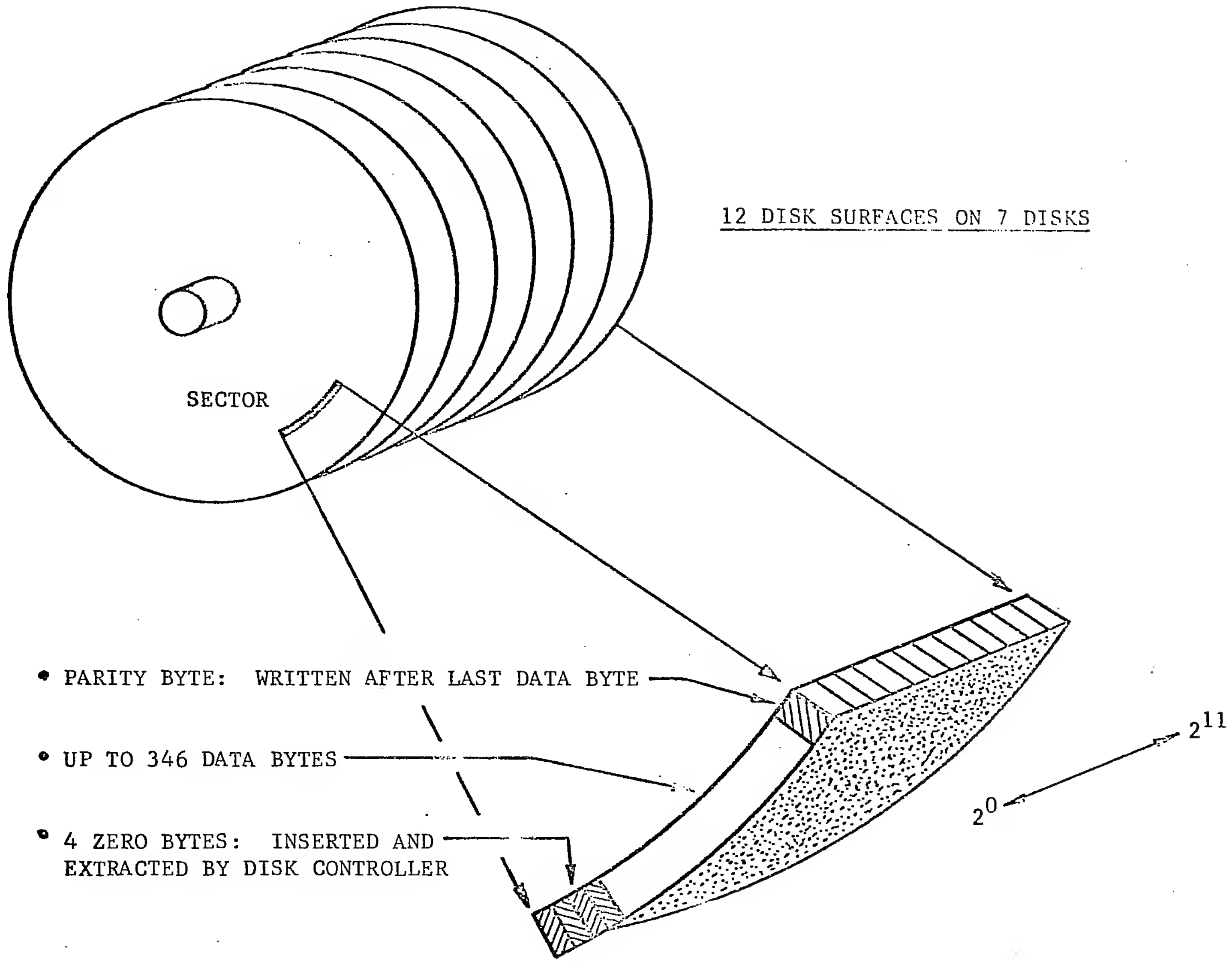

DISC ORGANIZATION
SECTOR 0 (OUTER)
SECTOR 1 (OUTER)
ZONE 1
ZONE 2
ZONE 3
ZONE 4
TRACK 0
TRACK 127
0
1
INNER SECTORS
REFERENCE MARK
GROUP SWITCH GAP
HEAD
HEAD BAR
ROCKER ARM

Figure 2



SECTOR FORMAT: 6603 DISK FILE

Figure 3

and only one sector is written at a time (i.e., data is recorded in physical records of one sector).

A reference mark on the disks containing the timing tracks defines the beginning of sector 0 in all four zones. Beyond this point, the starting point of sectors in the two inner zones does not coincide with the starting point of sectors in the two outer zones (see figure 2). The clock surfaces contain timing tracks for each zone. As the disk rotates, one of these timing tracks (depending on which head group is selected) drives a cell counter. This counter in turn triggers a sector counter. Both counters are initialized when the reference mark is detected. The cell counter is incremented as the timing track is read: When it reaches a count of 351, it is reset and the sector count advanced. The controller compares the sector number specified in a read or write function code: When equality is obtained, the read or write operation is initiated. The contents of the sector counter appear in the low-order 7 bits of the status response.

6603 Disk File: Timing Considerations

The rotational speed of the disk is approximately 900 revolutions per minute, corresponding to a revolution time of about 66 milliseconds. The time required to read or write a byte is approximately 1.4 microseconds on the two outer zones and 1.8 microseconds on the two inner zones. In the outer zones, then, a sector passes under the heads every 490 microseconds. It requires a minimum of 325 microseconds to transfer the 64 central memory words in a sector from peripheral processor memory to central memory, and, because of memory and pyramid conflicts, will probably require longer. A single peripheral processor cannot maintain a continuous data flow between consecutive sectors on the disk and central memory.

If the programmer wishes to read or write in a given sector, he simply issues the appropriate function code and, when the sector comes under the heads, the operation is initiated. The programmer may prefer to minimize the time spent waiting for this sector by sensing (via a status request) the position of the disk. Timing considerations make

it impossible to sense for a given sector and then initiate an operation in that sector: If one wishes to read or write sector N, then sector N-2 should be sensed in order to assure that a revolution will not be lost.

There are two types of delays which are of concern to the disk programmer. One of these is the positioning delay: The time required to move the heads to a new track. When a track select function has been received by the disk controller and positioning initiated, a delay determined by counting ~~two~~ 4 reference marks is provided to permit the head assembly to stabilize. Thus, depending on when positioning is initiated, up to ~~133~~ 266 milliseconds may be required. During positioning, a status request will receive a "NOT READY" reply.

The second type of delay is the switching delay encountered when a different head group is selected. When head group switching is initiated, the controller provides a one millisecond delay to allow the circuits to stabilize: Furthermore, reading or writing cannot be initiated until a reference mark is detected. Thus, depending on when the head group select function is issued, up to 66 milliseconds may be required for head group selection.

Between the last sector in a track (sector 127 in the outer zones, sector 99 in the inner zones) and the first sector (sector 0) on that track is an area called the group switch gap (see figure 2). This area is approximately equivalent to three sectors in size. It is provided to accommodate the minimum 1 millisecond switching delay. A programmer can thus read or write the last sector in a track, select a new head group, and read or write sector zero of the new track without incurring a delay.

The function code for head group selection is 160X, where X is the head group number (0-7). It is possible to vary the second octal digit in this function code (normally zero) from 1 to 7: In doing so, the manner in which the data signals from the disk are sampled is varied. Use of the feature is reserved for error routines.

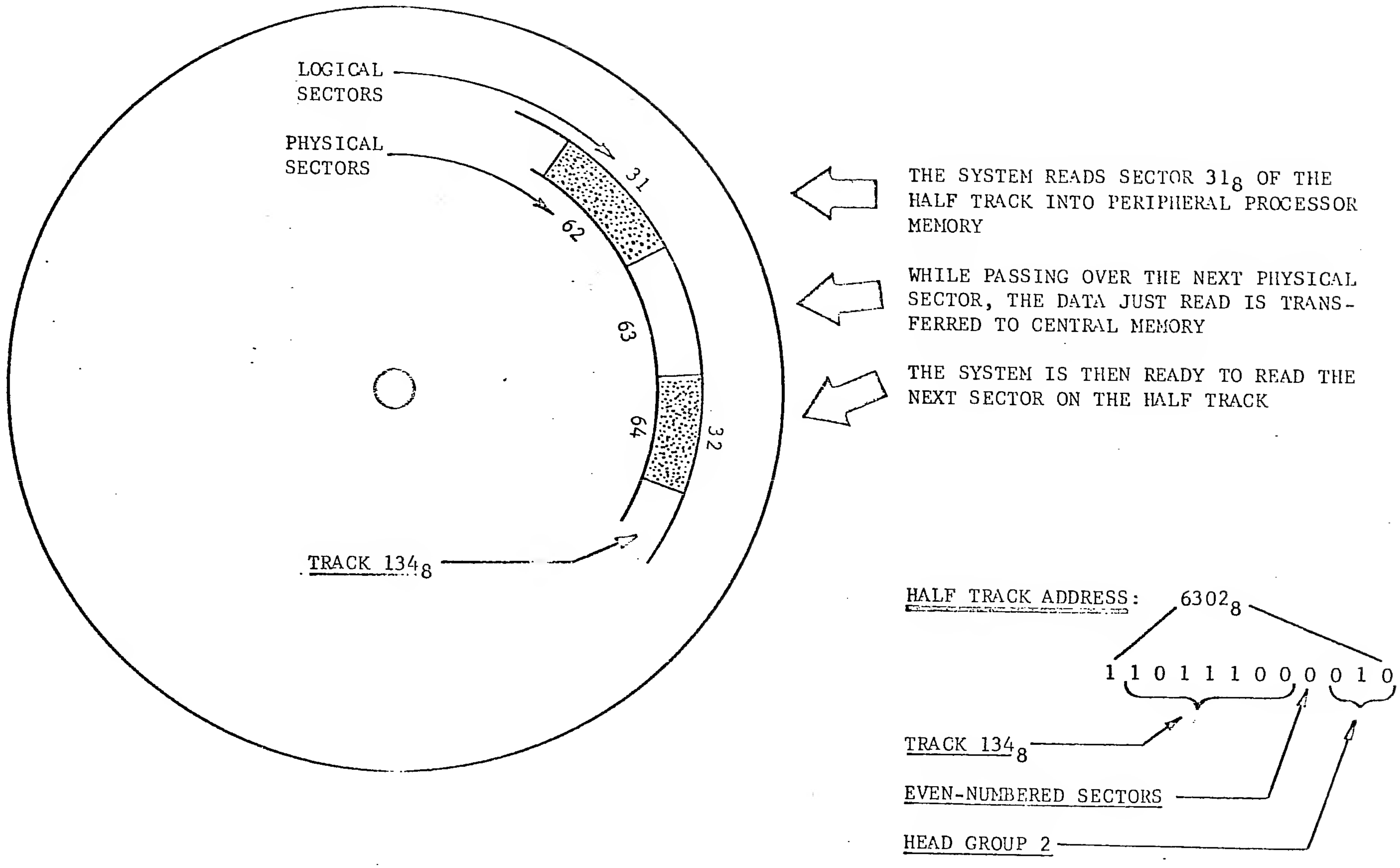
6603 Disk File: Data Capacity

There are 128 physical positions of the heads: At any one position, a track may be accessed by selecting one of eight head groups. Thus, the disk has a total of 8 x 128 = 1024 tracks. Of the eight head groups, four cover inner zones and four cover outer zones. In the inner zones, there are 100 sectors per track: In the outer zones, there are 128 sectors per track. Therefore, 512 tracks each contain 100 sectors while the other 512 tracks each contain 128 sectors. The disk file thus contains 116, 736 sectors. In normal use, up to 64 central memory words are recorded in a sector. The capacity of the 6603 disk file is thus approximately 7.5 million central memory words.

Chippewa Operating System Disk Usage

As we have seen, a single peripheral processor cannot maintain a continuous data flow from consecutive disk sectors to central memory. Therefore, the Chippewa Operating System uses a half track scheme in its disk operations. A half track is composed of either the odd-numbered or the even-numbered sectors in a track. In a disk operation, the system reads or writes alternate sectors, transferring data to or from central memory while passing over the intervening sector. Since the disk contains 1024 physical tracks, the equivalent half track capacity is 2048. The allocation of half tracks is controlled by MTR: disk write routines obtain half track addresses from MTR via the Request Track function. MTR maintains a table called the Track Reservation Table (TRT) which contains an entry for each half track on a disk. On receipt of the Request Track function, MTR searches the table for an unassigned half track, and returns the half track address to the requestor in the upper byte of the Message Buffer. If no half track is available, a zero address is returned to the requestor. A half track is never split between files: thus, the half track is the smallest unit of storage allocated on the disk.

The format of the half track address, and its relationship to physical disk addresses, is illustrated below.



HALF TRACK USE: AN EXAMPLE

Figure 4

```
1XXXXXXXXXXX
 |     ||
 |     |└──── head group number (0-7₈)
 |     └───── "1" if odd sectors, "0" if even sectors
 └─────────── track number (0-177₈)
```

Sector numbers maintained by the system (such as the Current Sector in an FST entry) are logical sector numbers, and refer to a sector within a half track. In the outer zones, sectors within a half track are numbered $0\text{-}77_8$: In the inner zones, sectors within a half track are numbered $0\text{-}61_8$. To convert a logical sector number to a physical sector number, the system shifts the logical sector number left one place and inserts the 2^4 bit from the half track address into the low-order bit position. For example, consider logical sector 77_8 (63_{10}) in a half track composed of the odd-numbered sectors in a physical track. In this case, the 2^4 bit of the half track address will be a "1". By shifting the logical sector left one place and inserting the "1" bit from the 2^4 bit position of the half track address, we obtain 177_8 (127_{10}) for the physical sector number. For the remainder of our discussion, a reference to "sector number" will refer to the logical sector number unless otherwise described.

For files recorded on the disk, the physical record is, of course, the sector. A logical record may be composed of several sectors. The format of the physical record is shown in figure 5. 502_8 bytes are always written in each sector. The first two bytes written are control bytes: the remaining 500_8 bytes are data bytes. Control byte 2 contains the number of useful central memory words in this sector: If control byte 2 contains 100_8, all 500_8 bytes in this sector contain useful information. A sector in which control byte 2 contains less than 100_8 is called a short sector, and is interpreted as a record mark. A logical record may comprise several full sectors, but is always terminated by a short sector. If the data to be recorded as a logical record is a multiple of 100_8 CM words, the system will write, as the record mark, a sector in which control byte 2 contains zero.

Control byte one points to the next physical record in this file. If the next sector is on the same half track, then this byte contains the

CONTROL BYTE 1	CONTROL BYTE 2	320 BYTES ALWAYS	WRITTEN

NUMBER OF USEFUL CM WORDS IN THIS SECTOR

POINTER TO NEXT SECTOR

- SECTOR NUMBER (0 - 77_8) IF ON SAME HALF TRACK
- HALF TRACK NUMBER IF ON ANOTHER HALF TRACK

CONTROL BYTE 1	CONTROL BYTE 2	RECORD
NON-ZERO	100_8	"FULL" SECTOR: PART OF A LOGICAL RECORD
NON-ZERO	NON-ZERO, $<100_8$	"SHORT" SECTOR: PART OF A LOGICAL RECORD; RECORD MARK
NON-ZERO	ZERO	"SHORT" SECTOR: RECORD MARK
ZERO	ZERO	FILE MARK

DISK FILE PHYSICAL RECORD FORMAT

Figure 5

number of that sector. If the next sector is on another half track, then this byte contains the half track address for that half track. (The file would be continued beginning with sector zero of the new half track.)

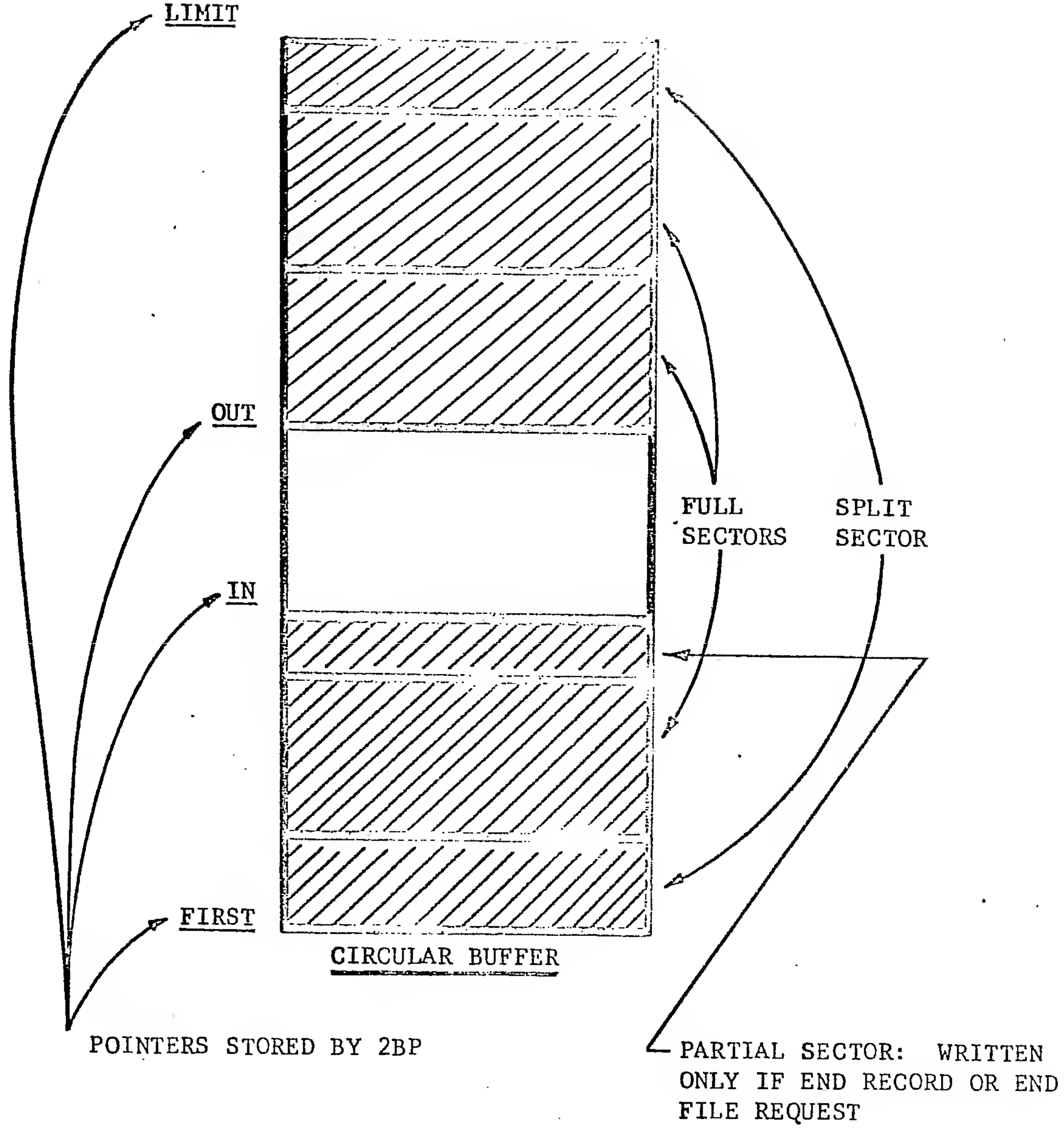
At the end of each write operation, the system writes a file mark. The Current Sector byte of the FST entry is not incremented to reflect this file mark sector, so the effect is equivalent to writing a file mark and backspacing over it. On the disk, a file mark is a sector in which both control bytes contain zero.

The Disk Write Overlay, 2WD

Disk write requests by users are executed by CIO's overlay 2WD. This overlay is also used by 1LJ and 1LT in loading jobs on the disk. Before calling 2WD, CIO calls the 2BP overlay to check the legality of the buffer parameters FIRST, IN, OUT, and LIMIT. After checking these parameters, 2BP searches the File Name Table for the file name specified in the CIO call (i.e., in the first word of the argument list). When found, 2BP stores the address of the corresponding FST entry. Should the file name not be found in the FNT, 2BP constructs an FNT entry for this file. Finally, 2BP clears the 2^0 bit in the buffer status byte of the FST entry to reserve the file.

CIO then calls 2WD. (Refer to the flow chart on page A-1.) 2WD reads the FST entry for the file and extracts the equipment number from byte one. The equipment number is added to the EST base address, and the EST entry read. The channel number from byte 2 of the EST entry is then inserted in the appropriate I/O instructions.

The output data in the circular buffer may appear as a contiguous block, or may wrap around the buffer, as illustrated in figure 6. In computing the total number of sectors in the circular buffer, then, the 2WD routine first subtracts OUT from IN. If the difference is positive, then this difference is the total number of words to be written, and 2WD shifts off the lower six bits of this word count in order to obtain the equivalent number of sectors. If OUT-IN is negative, the value of LIMIT is added to the difference and FIRST subtracted to obtain the



BUFFER PARAMETER PROCESSING

1. COMPUTE TOTAL NUMBER OF WORDS IN OUTPUT AREA

2. COMPUTE TOTAL NUMBER OF SECTORS IN OUTPUT AREA BY SHIFTING TOTAL WORD COUNT RIGHT 6 PLACES

3. COMPUTE NUMBER OF WORDS BETWEEN OUT AND LIMIT

4. COMPUTE NUMBER OF SECTORS BETWEEN OUT AND LIMIT BY SHIFTING OUT-LIMIT WORD COUNT RIGHT 6 PLACES

5. EXTRACT LOW-ORDER 6 BITS OF OUT-LIMIT WORD COUNT: THIS GIVES THE NUMBER OF WORDS IN THAT PART OF THE SPLIT SECTOR BETWEEN OUT AND LIMIT

6. SUBTRACT NUMBER OF WORDS COMPUTED IN (5) FROM 100_8 TO GET NUMBER OF WORDS TO BE READ FROM THE BUFFER BEGINNING AT FIRST IN ORDER TO COMPLETE THE SPLIT SECTOR

7. SET UP INSTRUCTIONS FOR PROCESSING THE SPLIT SECTOR

CIRCULAR BUFFER PARAMETER PROCESSING - 2WD OVERLAY

Figure 6

total word count and, from that, the equivalent number of sectors.

Regardless of whether the data is contiguous or wraps around the buffer, 2WD proceeds on the assumption that the data does wrap around, and proceeds to compute the values needed to process the wraparound case. The steps involved are listed in figure 6. These values, although always computed, are not required in the contiguous case: in either case, the terminal path is entered when the total sector count is reduced to zero. By computing these values regardless of whether the data is contiguous in the buffer or wraps around the buffer, computations during the period when the disk is actively in use are reduced.

Next, 2WD picks up the channel number from the EST entry and requests reservation of that channel from MTR. The Current Track byte of the FST entry for this file is then examined. If this byte is zero, then this file has not previously been used. A half track assignment is requested from MTR: MTR returns a half track address to the requestor in byte one of the first word in the message buffer. If no half track is available, MTR will return a zero byte to the requestor: 2WD then inserts an error message in the dayfile and aborts the control point after dropping the channel reservation. 2WD now has the address of the half track where the next operation is to be performed, and proceeds to position the disk to this half track. This half track address is compared with byte 2 of the TRT pointer word for this disk, and repositioning or head group selection performed only if required. Byte 2 of the TRT pointer is then updated.

2WD next requests another half track assignment from MTR. This half track is a spare: by keeping it available, it is possible for 2WD to switch head groups within the group switch gap if this action should be required when the end of the current half track is reached.

The transfer of data from the buffer to the disk then begins. 2WD reads 100_8 words from central memory into peripheral processor memory, sets control bytes one and two, and then writes the completed sector to the disk. As each sector is written, the number of the sector is examined to determine if the end of the half track is reached. To do this, 2WD compares the sector number with byte 4 of the TRT pointer word (if head

group number = 0-3) or byte 5 of the TRT pointer word (if head group number = 4-7). These bytes contain the values 100_8 and 62_8, respectively.

If the end of the half track has been reached, 2WD positions the disk to the spare half track: again, the half track address is compared with byte 2 of the TRT pointer word and positioning or head group selection performed only if required. After initiating any repositioning which might be required, 2WD requests a spare half track from MTR.

2WD continues reading 100_8-word blocks from central memory and writing them to the disk until it recognizes that there is not enough data in the circular buffer for a complete sector. (Some part of a sector may still, however, remain.) 2WD then examines the buffer status contained in byte 5 of the FST entry to see if an end record was requested (2^4 bit = 1). If an end record was requested, 2WD writes a short sector to the disk. If any data remained in the circular buffer, it will be written in this short sector: otherwise, control byte 2 will simply be set to zero.

After the last data sector has been written to the disk, 2WD writes a file mark - a sector with both control bytes equal to zero. The Current Sector byte of the FST entry is not, however, incremented to reflect the writing of this file mark: the next write to this file will write over the file mark sector. After the file mark has been written, 2WD requests MTR to drop the spare half track assignment and to release the channel reservation.

If no end record function was requested, 2WD simply updates the OUT pointer before returning control to CIO: There may still be some data in the circular buffer. If an end record function was requested, no data remains in the buffer: 2WD therefore sets IN = OUT = FIRST to indicate that the buffer is empty.

When control is returned to CIO, CIO sets the 2^0 bit of the buffer status in the FST entry to 1 to indicate that the file is no longer in use, and sets the 2^0 bit of the buffer status in the calling program's argument list to 1 to indicate to the calling program that the operation has been completed.

The Disk Read Overlay, 2RD

Disk read requests by users are executed by CIO's overlay 2RD. This overlay is also used by 1DJ and 1TD. The processing performed by 2BP in this case is identical to that performed in the case of 2WD. On entry, 2RD reads the FST entry for the file, picks up the equipment number from byte one, and uses this number to obtain the EST entry. The channel number from the EST entry is then set in the I/O instructions.

2RD then proceeds to compute the number of sectors which can be loaded into the circular buffer. If there is not room for a full sector, control is returned to CIO. The data to be read may fit in the buffer in a contiguous block, or may wrap around the buffer. The computation of the values (total word count, total sector count, etc.) used in controlling the transfer of data to the buffer is performed in a manner similar to 2WD. Again, the wraparound case is assumed.

The Current Track byte of the FST entry is examined. If this byte is zero, the file has not been used before and so contains no data. 2RD sets the buffer status to indicate a file mark and returns control to CIO.

2RD requests a channel reservation from MTR and positions the disk to the half track address contained in the FST entry's Current Track byte. As in all disk routines, the half track address is compared with the disk position specified in the TRT pointer, and repositioning or head group switching performed only if necessary.

2RD then uses the Current Sector byte of the FST entry to construct the read function code, and reads the specified sector into peripheral processor memory. A status request is then issued, and the response is examined to determine if a parity error occurred. In the event of a parity error, the system rereads the sector three times; once using the normal sampling method and twice at varied sampling margins. If the parity error re-occurs in each of the rereads, 2RD inserts an error message in the dayfile and stops (via a UJN 0 instruction). Since the halt occurs without the disk channel being released, all system activity will shortly cease (if this disk is the

system disk, disk 0). A dead start load will be necessary to reinitiate processing.

If the read was successful, 2RD examines the high-order six bits of control byte one: if these bits are zero, then this control byte contains a sector number, while if these bits are non-zero, this control byte contains a half track number. In the latter case, 2RD positions the disk to the new half track address. While any repositioning or head group switching which might be required is in process, 2RD transfers the number of words specified in control byte 2 from peripheral processor memory to the circular buffer, and updates the values used in controlling the transfer. If the sector just read was a full sector (100_8 CM words of data), and if there is enough room in the circular buffer for another full sector, 2RD loops to read the next sector from the disk.

If the last sector read was a short sector, then the end of a logical record has been reached, and the buffer status is set to reflect a record mark. If the end of logical record has been reached, or if there is not enough room in the circular buffer for a full sector, 2RD requests MTR to release the channel reservation, updates the IN pointer in the calling program's argument list, and returns control to CIO. CIO updates the buffer status in the FST entry to release the file reservation, and updates the buffer status in the calling program's argument list to indicate that the operation has been completed.

If, after reading the last logical record in a file, the calling program issues another read to the file, the file mark will be read. The processing proceeds as described above: 2RD reads a sector whose address is specified in the Current Track and Current Sector bytes of the FST entry. Since control byte 2 is zero, 2RD recognizes this as a short sector, sets the buffer status to reflect a record mark, and releases the channel. 2RD then examines control byte one; since this contains zero, the file mark is recognized and the buffer status set accordingly before returning control to CIO.

The Backspace Disk Overlay, 2BD

Disk backspacing may take the form of a BCD backspace or, more commonly, a binary backspace. In either case, it is desired to backspace over a logical record, and it is assumed that any backspacing over logical records in the buffer has been done by the calling program. Backspacing over the physical records which may constitute a logical record is essentially a matter of backspacing over two sectors and then reading a sector.

2BD uses a subroutine to backspace over a sector. (See flow chart on page A-5.) This subroutine examines the Current Sector byte of the FST entry, and, if non-zero, subtracts one from this number and exits. This is equivalent to backspacing over one physical record (i.e., one sector). If the Current Sector number is zero, then the preceding physical record is on another half track. In this case, the subroutine stores the Current Track byte from the EST entry for this file, since it will have to search the file for a sector which has this half track address contained in control byte one.

The subroutine rewinds the file by picking up the Beginning Track byte from the FST entry. (Should the Beginning Track byte be equal to the Current Track byte, the subroutine exits, since this indicates that the system has backspaced over all physical records in this file.) After rewinding the file, the subroutine reads each sector in the file until it finds a sector with the desired half track address in control byte one. The number of this sector is then stored, and control returned to the calling routine. A backspace operation on a file of any size may take considerable time if it should become necessary to rewind the file and search forward.

A binary backspace on the disk consists of backspacing over two sectors (using the subroutine described above) and reading a sector until a short record is found, indicating the end of a logical record. 2BD sets the circular buffer pointers IN and OUT equal to FIRST, and returns control to CIO. CIO updates the buffer status in the FST entry and in the calling program's argument list before exiting.

It is also possible to issue a BCD backspace to the disk. For the disk, as for 1" tape (but not for 1" tape), a logical BCD record consists of a series of central memory words presumably containing display code data, terminated by a central memory whose low-order byte (byte 5) is zero.

The BCD backspace begins with the computation of the amount of data left in the buffer as a result of the last read. This quantity, referred to as D, is equal in IN-OUT if the data in the buffer is contiguous, or IN-OUT + LIMIT-FIRST if the data wraps around the buffer. This data was left in the buffer as a result of the last read, and may have been stored on the disk in several sectors. The system assumes that the calling program will backspace within the buffer, and so, before beginning a logical BCD record backspace on the disk, 2BD will backspace the disk a number of sectors equivalent to the amount of data contained in the buffer. This quantity is represented by D.

2BD therefore backspaces over a sector (by the same subroutine used in binary backspacing and described earlier) and reads that sector into peripheral processor memory. The sector length in control byte 2 is then compared with D: if less than D, then this sector is assumed to contain data which has already been read into the buffer. 2BD then decreases D by this amount, backspaces over this sector and the sector preceding it, and then reads a sector. The process of backspacing, reading, and reducing D is repeated until a sector is read whose length is greater than the present value of D: this sector could not entirely be part of the read data in the buffer, and so must be searched for a logical record. 2BD transfers this sector from peripheral processor memory to the circular buffer beginning at FIRST. If D is still non-zero, then part of this sector contains data residing in the buffer at the time the backspace was requested, and presumably has been searched by the calling program: 2BD therefore sets the OUT pointer to FIRST + sector length - D. At the same time, the IN pointer is set to reflect the transfer of the sector to the buffer.

2BD then searches each word in the buffer from OUT - 1 down to FIRST until a word with a zero low-order byte is found, indicating the end of a logical BCD record. When the end of the record is found, 2BD updates the IN and OUT pointers in the calling program's argument list, and

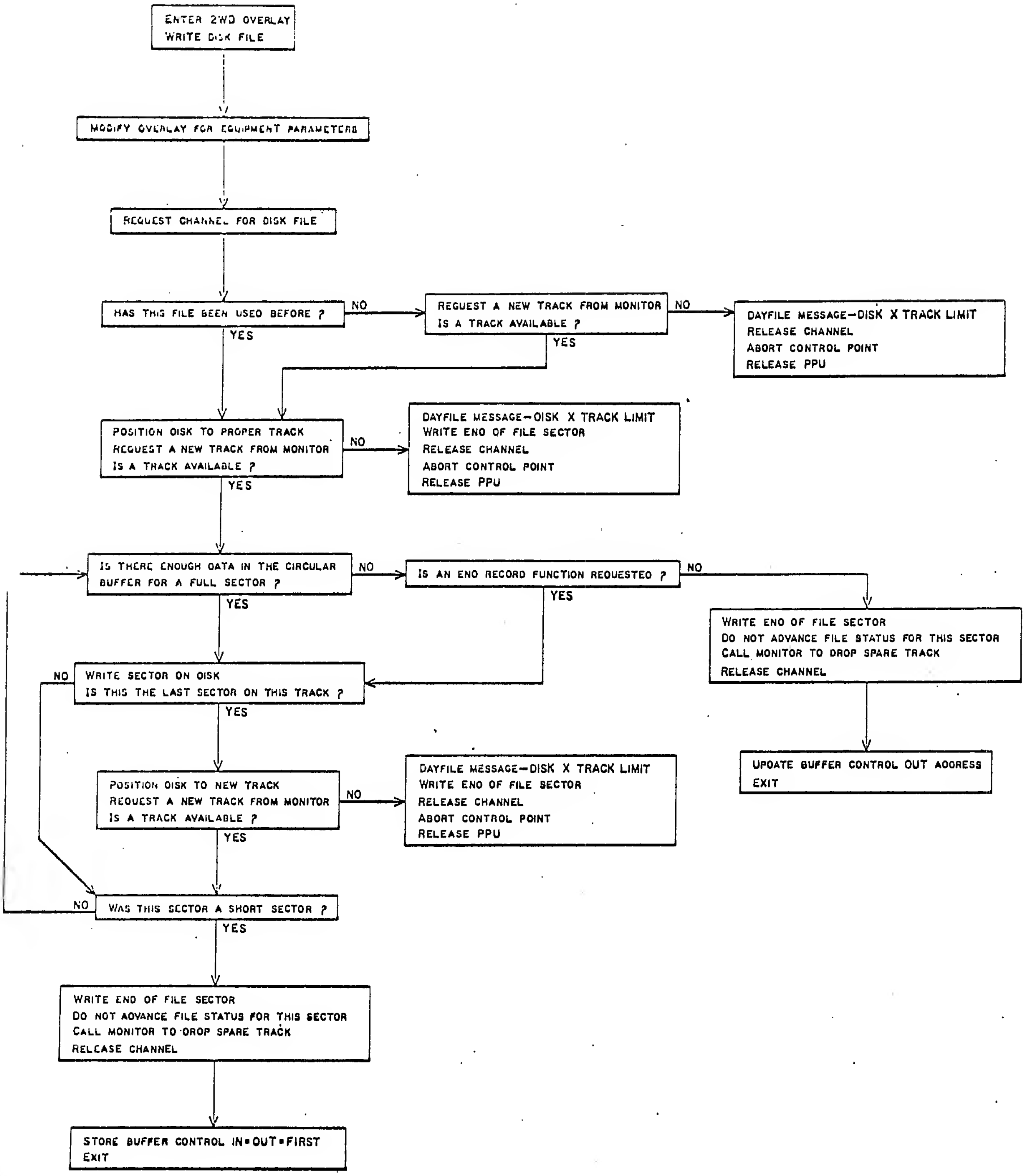
returns control to CIO. OUT now points to the first word following the end of the logical record. If no zero low-order byte was found, then 2ED backspaces two sectors and reads one, and then repeats the buffer search.

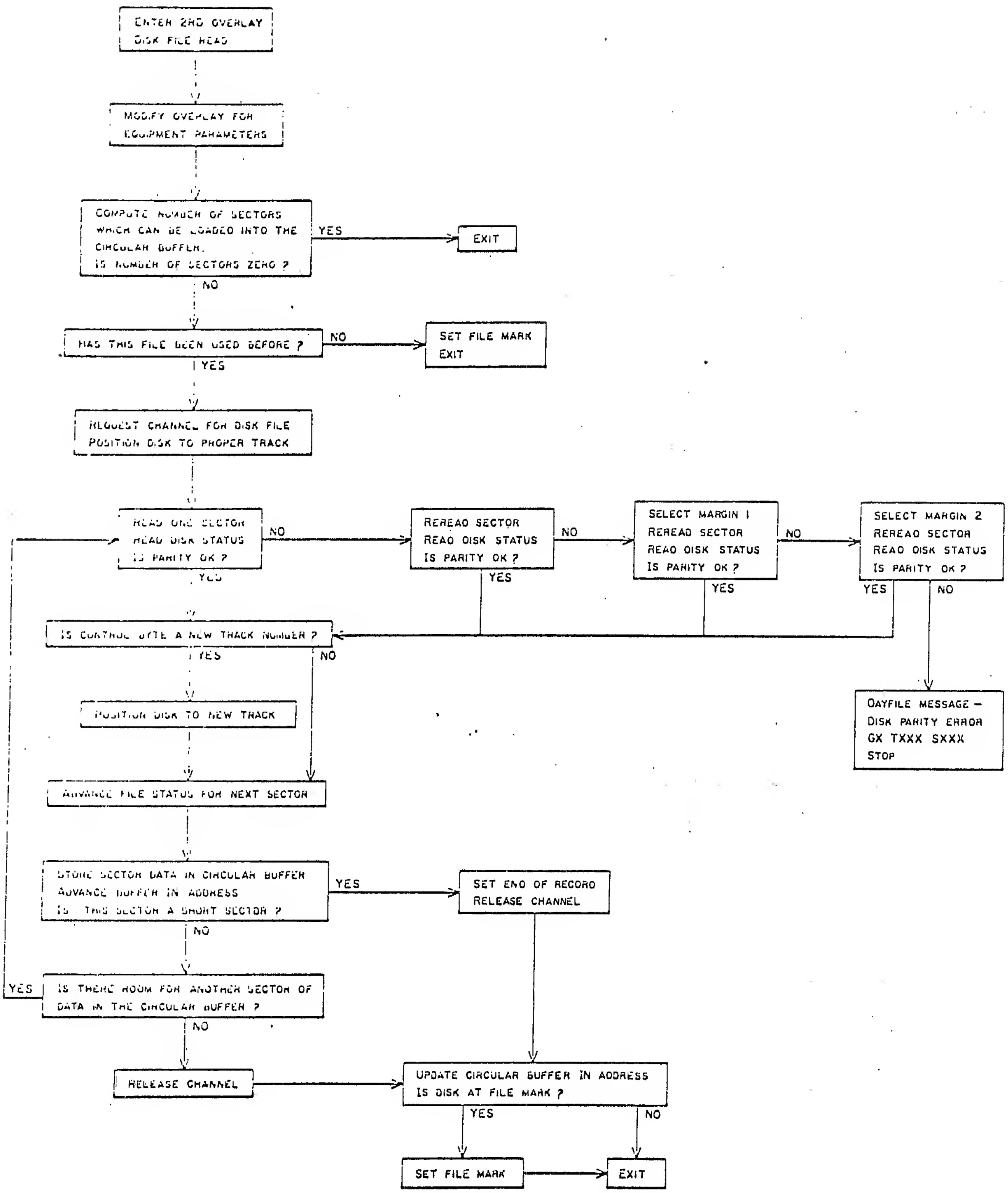
The Drop Track Overlay, 2DT

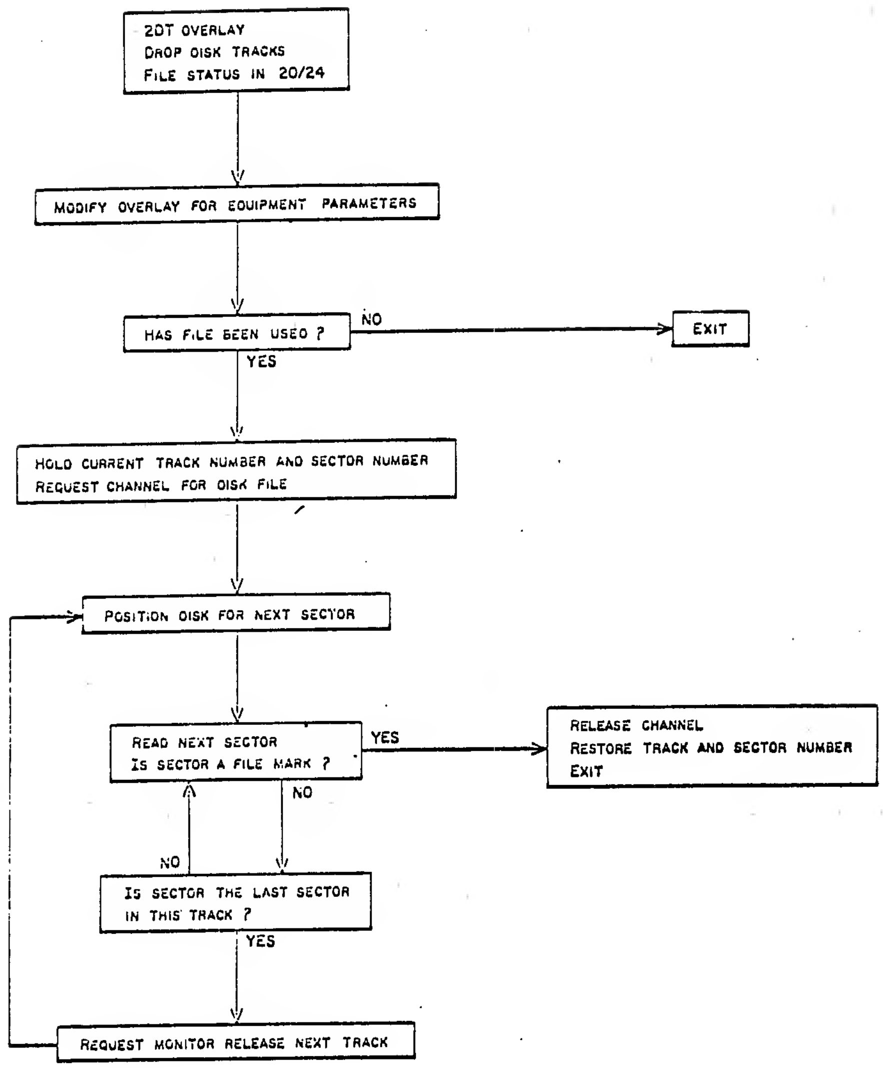
When CIO receives a disk write request, it first calls the 2BP overlay to check the legality of the buffer parameters and to search the FNT for the file name. CIO then reads the EST entry for this file, and examines the buffer status in byte 5. If the buffer status indicates that the last operation performed on this file was a read operation, then an overlay, 2DT, is called to drop the subsequent portion of the file. In effect, then, if some part of a file is read and it is then decided to write to that file, the remainder of the file is erased.

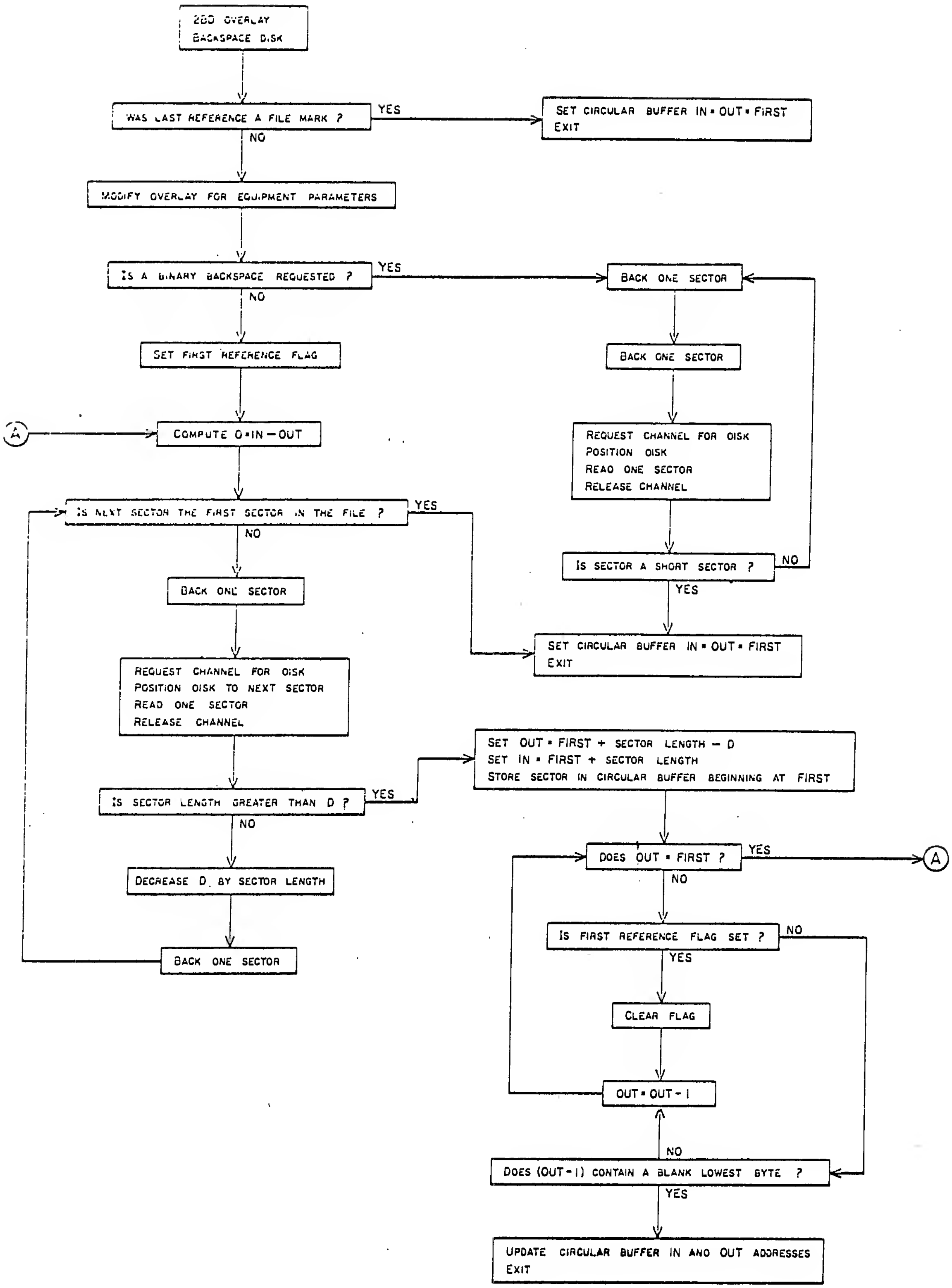
The flow chart for the 2DT overlay is shown on page A-3 of the attached flow charts. The routine picks up the Current Track byte and Current Sector byte from the FST entry for the file, and reads the sector at this address. If this sector is a file mark, 2DT returns control to CIO. If control byte one of this sector contains a half track address, 2DT requests MTR to drop this half track reservation. MTR then clears the bit in the Track Reservation Table corresponding to this half track address. 2DT positions the disk to this half track address and begins reading sectors until a file mark is found or the end of the half track is reached. The process of reading and dropping half tracks continues until the end of the file is reached.

At the end of a job, all local files associated with the job are dropped. For disk files, a process similar to that described above is required to release half track reservations. This is performed for 1AJ by the 2DF overlay. 2DF differs from 2DT in that 2DF drops files assigned to other equipment as well as those assigned to the disk, and 2DF drops all the half tracks reserved by a file, not just those following the half track specified in the Current Track byte of the FST entry. 2DF is also called by 1DJ and 1TD when printing files or writing files on tape.

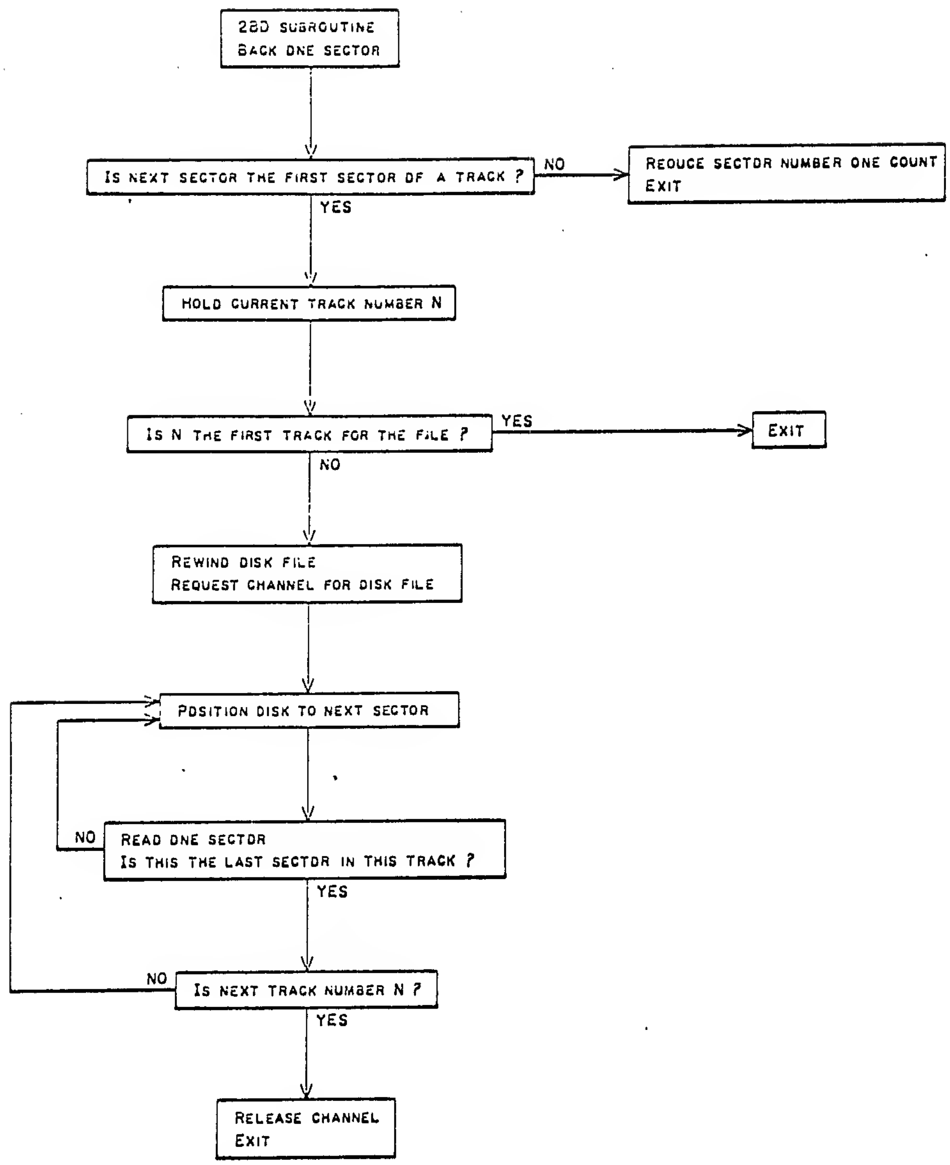

ENTER 2WD OVERLAY
WRITE DISK FILE
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
REQUEST CHANNEL FOR DISK FILE
HAS THIS FILE BEEN USED BEFORE ?
NO
YES
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
POSITION DISK TO PROPER TRACK
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
WRITE END OF FILE SECTOR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
IS THERE ENOUGH DATA IN THE CIRCULAR
BUFFER FOR A FULL SECTOR ?
NO
YES
IS AN END RECORD FUNCTION REQUESTED ?
NO
YES
WRITE END OF FILE SECTOR
DO NOT ADVANCE FILE STATUS FOR THIS SECTOR
CALL MONITOR TO DROP SPARE TRACK
RELEASE CHANNEL
UPDATE BUFFER CONTROL OUT ADDRESS
EXIT
NO
WRITE SECTOR ON DISK
IS THIS THE LAST SECTOR ON THIS TRACK ?
YES
POSITION DISK TO NEW TRACK
REQUEST A NEW TRACK FROM MONITOR
IS A TRACK AVAILABLE ?
NO
YES
DAYFILE MESSAGE—DISK X TRACK LIMIT
WRITE END OF FILE SECTOR
RELEASE CHANNEL
ABORT CONTROL POINT
RELEASE PPU
NO
WAS THIS SECTOR A SHORT SECTOR ?
YES
WRITE END OF FILE SECTOR
DO NOT ADVANCE FILE STATUS FOR THIS SECTOR
CALL MONITOR TO DROP SPARE TRACK
RELEASE CHANNEL
STORE BUFFER CONTROL IN=OUT=FIRST
EXIT


ENTER 2ND OVERLAY
DISK FILE READ
MODIFY OVERLAY FOR
EQUIPMENT PARAMETERS
COMPUTE NUMBER OF SECTORS
WHICH CAN BE LOADED INTO THE
CIRCULAR BUFFER.
IS NUMBER OF SECTORS ZERO ?
YES
EXIT
NO
HAS THIS FILE BEEN USED BEFORE ?
NO
SET FILE MARK
EXIT
YES
REQUEST CHANNEL FOR DISK FILE
POSITION DISK TO PROPER TRACK
READ ONE SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
SELECT MARGIN 1
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
NO
SELECT MARGIN 2
REREAD SECTOR
READ DISK STATUS
IS PARITY OK ?
YES
YES
YES
YES
NO
DAYFILE MESSAGE –
DISK PARITY ERROR
GX TXXX SXXX
STOP
IS CONTROL BYTE A NEW TRACK NUMBER ?
YES
NO
POSITION DISK TO NEW TRACK
ADVANCE FILE STATUS FOR NEXT SECTOR
STORE SECTOR DATA IN CIRCULAR BUFFER
ADVANCE BUFFER IN ADDRESS
IS THIS SECTOR A SHORT SECTOR ?
YES
SET END OF RECORD
RELEASE CHANNEL
NO
YES
IS THERE ROOM FOR ANOTHER SECTOR OF
DATA IN THE CIRCULAR BUFFER ?
NO
RELEASE CHANNEL
UPDATE CIRCULAR BUFFER IN ADDRESS
IS DISK AT FILE MARK ?
YES
NO
SET FILE MARK
EXIT


2DT OVERLAY
DROP DISK TRACKS
FILE STATUS IN 20/24
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
HAS FILE BEEN USED ?
NO
EXIT
YES
HOLD CURRENT TRACK NUMBER AND SECTOR NUMBER
REQUEST CHANNEL FOR DISK FILE
POSITION DISK FOR NEXT SECTOR
READ NEXT SECTOR
IS SECTOR A FILE MARK ?
YES
RELEASE CHANNEL
RESTORE TRACK AND SECTOR NUMBER
EXIT
NO
NO
IS SECTOR THE LAST SECTOR
IN THIS TRACK ?
YES
REQUEST MONITOR RELEASE NEXT TRACK



280 OVERLAY
BACKSPACE DISK
WAS LAST REFERENCE A FILE MARK ?
YES
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
NO
MODIFY OVERLAY FOR EQUIPMENT PARAMETERS
IS A BINARY BACKSPACE REQUESTED ?
YES
BACK ONE SECTOR
NO
SET FIRST REFERENCE FLAG
BACK ONE SECTOR
A
COMPUTE D = IN − OUT
REQUEST CHANNEL FOR DISK
POSITION DISK
READ ONE SECTOR
RELEASE CHANNEL
IS NEXT SECTOR THE FIRST SECTOR IN THE FILE ?
YES
NO
IS SECTOR A SHORT SECTOR ?
NO
BACK ONE SECTOR
YES
SET CIRCULAR BUFFER IN = OUT = FIRST
EXIT
REQUEST CHANNEL FOR DISK
POSITION DISK TO NEXT SECTOR
READ ONE SECTOR
RELEASE CHANNEL
SET OUT = FIRST + SECTOR LENGTH − D
SET IN = FIRST + SECTOR LENGTH
STORE SECTOR IN CIRCULAR BUFFER BEGINNING AT FIRST
IS SECTOR LENGTH GREATER THAN D ?
YES
NO
DOES OUT = FIRST ?
YES
A
DECREASE D BY SECTOR LENGTH
NO
IS FIRST REFERENCE FLAG SET ?
NO
BACK ONE SECTOR
YES
CLEAR FLAG
OUT = OUT − 1
NO
DOES (OUT − 1) CONTAIN A BLANK LOWEST BYTE ?
YES
UPDATE CIRCULAR BUFFER IN AND OUT ADDRESSES
EXIT



28D SUBROUTINE
BACK ONE SECTOR
IS NEXT SECTOR THE FIRST SECTOR OF A TRACK ?
NO
REDUCE SECTOR NUMBER ONE COUNT
EXIT
YES
HOLD CURRENT TRACK NUMBER N
IS N THE FIRST TRACK FOR THE FILE ?
YES
EXIT
NO
REWIND DISK FILE
REQUEST CHANNEL FOR DISK FILE
POSITION DISK TO NEXT SECTOR
NO
READ ONE SECTOR
IS THIS THE LAST SECTOR IN THIS TRACK ?
YES
NO
IS NEXT TRACK NUMBER N ?
YES
RELEASE CHANNEL
EXIT